Peer-to-peer Deep Learning

Robert Šajina

Faculty of Informatics, Juraj Dobrila University of Pula
robert.sajina@unipu.hr

***Abstract*—A new and fast-emerging paradigm of Federated Learning is enabling collaborative learning of deep neural networks on the edge devices by orchestrating the learning process from a central point. However, this centralization often imposes a communication bottleneck and raises privacy concerns requiring a more decentralized approach. In decentralized systems, edge devices mutually collaborate in a peer-to-peer manner. Decentralized or peer-to-peer collaborative learning techniques have increasingly attracted researchers' interest in recent years. Several decentralized approaches are already proposed. However, it is important to understand and analyze their properties. For a decentralized network of smart agents, it is important that the learning process is communication-efficient, tolerant of faulty or malicious agents, and protects agents' data privacy. This paper analyzes these properties for some of the latest proposed approaches.**

***Index Terms*—peer-to-peer, gossip averaging, decentralized learning, neural network, machine learning**

I. Introduction

With an increasing amount of connected edge devices such as mobile phones, tablets, and laptops, a vast amount of valuable but mostly private data can be used to train machine learning models, often in the form of Deep Neural Networks (DNNs). There is a growing research interest in utilizing edge devices' local data. A centralized approach would require pooling all edge devices' local data and training one model. However, that would violate the data privacy of edge device owners. Instead, efforts were made to utilize edge devices' computational capabilities to train a model locally and exchange this model collaboratively with other devices. This way, local data never leaves the device. An edge device can be considered as any device containing memory, computing power, internet access, and potentially valuable data that can be used as a model training dataset. These components can be utilized to perform model training on the device using the local device training data. Collaborative learning aims to ensure data privacy by only sharing the model with the outside world while keeping the data private. From hereafter, we refer to an edge device as an agent and as a node if considered as a part of network topology. We may also use these terms interchangeably. Each agent contains local data, which is private and never shared, a machine learning framework, such as Tensorflow [1], or PyTorch [2], and a pre-trained or randomly initialized machine learning model (see figure 1). The model architecture and parameters are, in general, specified upfront by the learning process.

A collaborative approach between agents may be based on a *parameter-server* paradigm or may be decentralized. Studies were conducted in both settings to address the data imbalance between agents. Local data between agents may have similar or different data distributions. Considering the MNIST [3] dataset of handwritten digits, an example of a balanced dataset is that all agents own a similar number of examples for each digit (similar distribution). Contrary, if some agents only own examples of odd digits and others own only even digits, then the distribution between agents' datasets is different. We formalize this balanced and unbalanced data setting in Section II. Section II also describes the differences between *parameter-server* and decentralized approaches, which are then discussed in more detail in Sections III and IV, respectively.

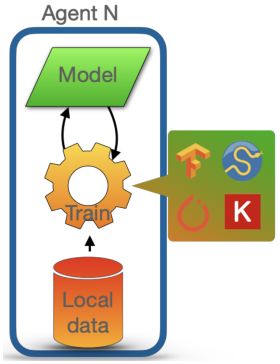

Fig. 1. Agent contains local data, machine learning framework and a model which is trained by the ML framework.

II. Background

The agent's local data consists of any data collected by the agent. This data includes images, texts, sensors, locations data, etc. Since agents are all individual devices, they generate or possess different data [4] also known as non independently or identically distributed (non-IID). Non-IID is the opposite of the IID case, where the distribution of data samples between agents is identical. Having IID data at the agents means that each batch of data used for an agent's local model update is statistically identical to a uniformly drawn sample (with replacement) from the entire training dataset, which is the union of all local datasets of the agents.

Consider a dataset of numeric values. A sample of size n consist of n random values: $\{X_1, X_2, ..., X_n\}$. This sample is IID if the random values have the following properties:

Independent: The random values $X_1, X_2, ..., X_n$ are independent, meaning that the occurrence of any X_i does not depend on any other X_j.

Identically Distributed: The random values $X_1, X_2, ..., X_n$ are from the same population and thus have the same distribution or cumulative distribution function (CDF) F:

$$F_{x_1} = F_{x_2} = ... = F_{x_n} = F_x$$

Training on IID data is easier and faster because all model updates converge toward one global solution. Oppositely, when agents have non-IID data, one global model solution is not optimal for all agents, making this case a challenging learning problem. Unfortunately, non-IID data is a common occurrence [5, 6], where the edge device's data differ in size, label mapping, etc., posing a challenge for collaborative training of the NN models. Culture, language, and usage patterns are just some factors that influence these data differences.

An agent can only train its model using its data. However, it is known that Deep Learning techniques produce better results with large quantities of data [7, 8]. Even though agents, in general, do not share data, exchanging model (or model parameters/weights) between agents (or a server) helps a model "see" more training data, making the model more accurate and robust. This vital collaboration between agents can either be guided by a central server or organized in a decentralized manner. In a decentralized system, the agent communicates with other agents in the network. These neighboring agents in the network are often referred to as peers. Each agent is responsible for its learning process, including communicating with peers and receiving and processing incoming messages. If the training process is orchestrated by a *parameter-server*, the agent, in general, communicates only with the server. *Parameter-server* learning system offers more control over the learning process and validation but imposes a communication bottleneck since all the communication must go through a central server computer (there can also be multiple server computers). A decentralized learning system nullifies this concern but presents a synchronization problem between agents. In this paper, we are focused on decentralized systems, but we still present some of the main approaches taken in a *parameter-server* paradigm.

III. Federated Learning

One of the fastest emerging and most commonly researched collaboration approaches is Federated Learning (FL) [9]. Federated Learning is a *parameter-server* paradigm used for orchestrating the process of training a global model using decentralized agents' data. The overview of the process is shown in figure 2. In each training round, the server selects a subset of currently online agents and sends them a copy of the current global model. Selected agents then train the global model on their local data for a specified number of iterations, utilizing their local computation capabilities and local (private) data. After an agent has finished training the model, it sends the model (or its delta) back to the central server. When most agents submit their updates to the server, the server proceeds with model averaging and creates a new global model. This process repeats until global model convergence.

Since the real-world agent datasets are mostly non-IID, efforts were made to improve the FL in that setting. Works such as [10, 11] use a public dataset to tackle the non-IID data setting. Recently, FedMD [10] has been proposed to enable Federated learning for heterogeneous models on non-IID data. This approach transfers knowledge to the server through model distillation [12]. Model distillation is a knowledge transfer technique that enables training a small model using a more knowledgeable larger network (see figure 3). For more details on different knowledge transfer approaches, refer to [13]. FedMD method presumes a public dataset similar to agents' private data. Each agent shares class scores on a public dataset with the server; the server then averages received scores and distributes updates to agents. These agents then train their models on a public dataset and private data. This cycle repeats a specified amount of times. Training on a public dataset (which may be large) is not feasible for agents. Furthermore, this means that agents would need to download this dataset for training purposes. In [11], a similar approach is proposed. An unlabeled public dataset is assumed, and instead of returning a model to the server, agents send their predictions of the public dataset. The server aggregates these predictions using a custom aggregation algorithm [14]. Agents then update their local models using their respectable data and the received updates from the server.

Approaches that adopt a similar algorithm as in [10, 11] are not feasible for the real world, as they have too many presumptions and communication restrictions. However, other approaches can be adopted when solving this problem [15]: (I) personalization, (II) multi-task learning, and (III) meta-learning.

On-device personalization of the previously learned global model implies performing a specific number of training iterations after the training is finished. This process can improve an individual agent's model accuracy [16]. On-device personalization is a sensitive operation since too much personalization will cause model overfitting, which degrades model accuracy. If done right, such an optimization method greatly improves the accuracy of local models, which deviate the most from the global model [17].

Each agent's problem is considered a separate task in a multi-task setting. The goal of the training process is to train one model per task [15, 18–23]. The relationship between agents' local datasets and learning tasks can be reconsidered as observing points on a spectrum between a single global model and different models for every agent. In multi-task learning, a subset of agents representing a task can be chosen explicitly (e.g., based on geolocation, agent device or user characteristics, etc.) or based on a learned clustering of the connected agents.

Meta-learning is another promising approach for studying personalization and non-IID data. This approach aims to meta-learn (train) a global model that can be used as a starting point for training a model adapted to a given task. Studies such as [24–26] show that this setting is a relevant framework to model the personalization objectives for FL.

IV. Decentralized learning

The learning outcome of *parameter-server* may be a single global model or multiple personalized models (if applying

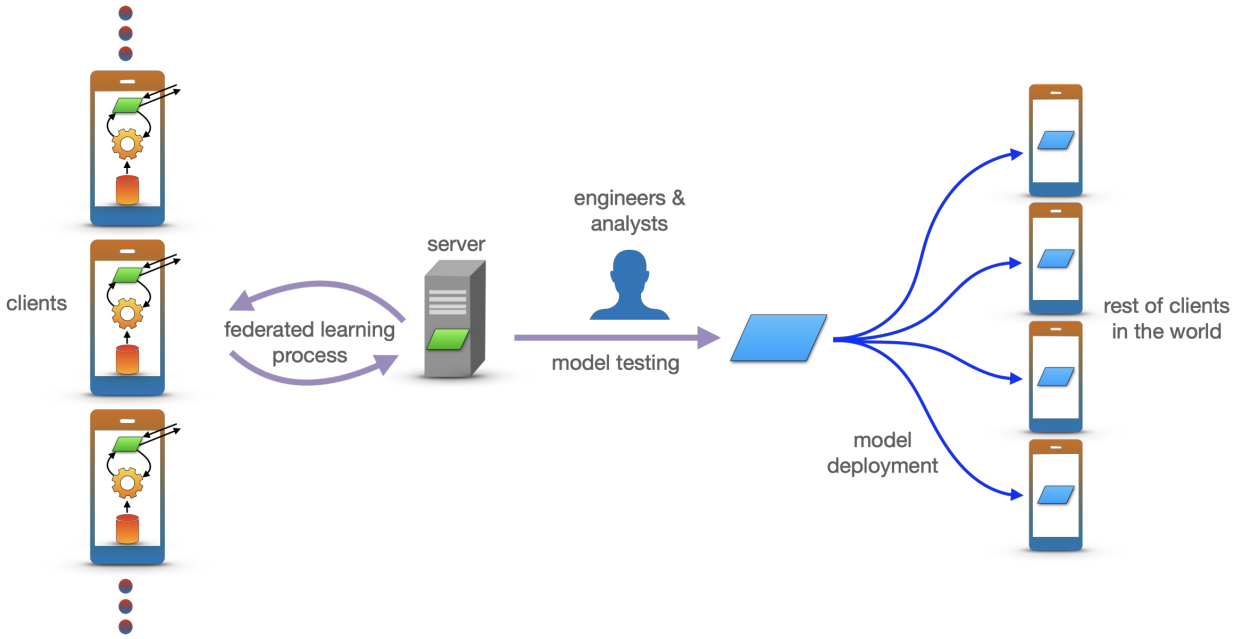

Fig. 2. Lifecycle of Federated Learning process and FL-trained global model. In each training round, the server chooses a subset of the online agents and sends them a copy of the current global model. Agents perform local training steps on the received model before returning the model to the server. This cycle is repeated for a finite number of rounds and, as a result, produces a robust global model. This model is then deployed as a part of an application and used by the users during regular application usage.

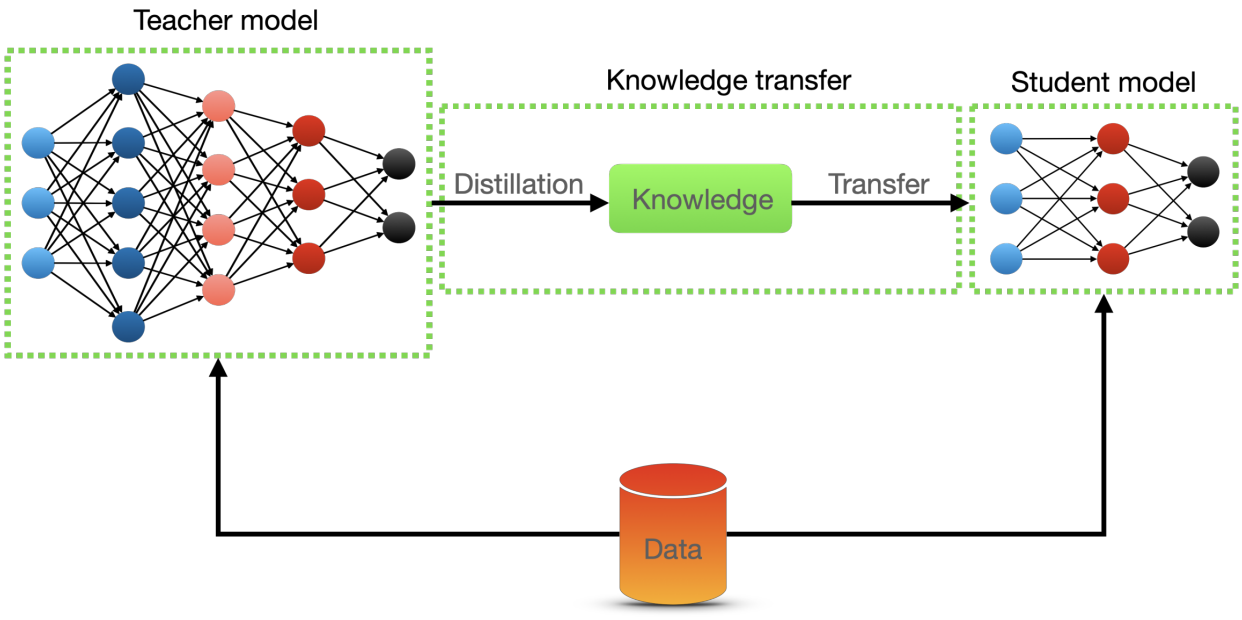

Fig. 3. Abstract teacher-student technique for knowledge distillation.

some of the mentioned techniques). The same can be considered in a decentralized system. Agents can collaboratively train one model or selfishly try to maximize the model's accuracy for their purpose (on their dataset), which we can consider as personalized models. Before analyzing these approaches, a detailed description of decentralized prerequisites is studied.

A. Network topology

The main difference between *parameter-server* and a decentralized system is that in a decentralized (or peer-to-peer) system, agents must organize the communication themselves. This main difference exposes various problems such as agent synchronization, agent trust, and data privacy.

Communication links between agents can be captured through a communication graph $G = (\llbracket N \rrbracket, E, W)$ where $\llbracket N \rrbracket = \{1, ..., N\}$ is a set of all nodes in the network, $E \in \llbracket N \rrbracket \times \llbracket N \rrbracket$ is the set of edges, and $W \in \mathbb{R}^{N \times N}$ is a nonnegative weighted matrix. Weight of edge $(i, j) \in E$ is given by W_{ij} with the convection $W_{ij} = 0$ if $(i, j) \notin E$ or $i = j$. An agent i only sends messages to agent j if $W_{ij} > 0$, which means that agent i communicates with peers $N_i = \{j : W_{ij} > 0\}$ without knowledge of other peers in the network, and operates without synchronisation with non-connected peers ($W_{ij} = 0$).

The resulting graph matrix shows the connection between agents. Consider the next matrix $W \in \mathbb{R}^{4 \times 4}$:

$$W = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Column indices of matrix W correspond to sending nodes, while row indices correspond to receiving nodes. For the node with index 0, Node_0, sending indices (column) are $\{0, 1, 0, 1\}$, meaning that Node_0 sends its message to nodes Node_1 and $Node_3$. Row at index 0 shows that Node_0 receives messages from both Node_1 and Node_3. This matrix example demonstrates an undirected/symmetric communication graph. The opposite of this is directed/asymmetric communication, in which an agent can send messages to a set of neighbors but receive messages from a completely different set of neighbors. In a directed graph, an agent has in-neighbour if $(i, j) \in E$ and out-neighbor if $(j, i) \in E$. Out-neighbor set represents the agents to which the agent sends the messages, while the in-neighbor set represents the agents from which messages are received. The number of in and out neighbors is, in general identical.

Commonly, the mixing matrix is comprised of fractions rather than integers. These fractions may signal the trust between agents or are simply a method of a decentralized approach to control the contributions of each agent. Consider the next matrix $W_m \in \mathbb{R}^{4 \times 4}$:

$$W_m = \begin{pmatrix} 1/2 & 0 & 0 & 1/3 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 1/2 & 1/2 & 1/3 \\ 0 & 0 & 1/2 & 1/3 \end{pmatrix}$$

In the mixing weights matrix, sending or receiving agents use these fractions to weigh individual agents' contributions depending on the approach. The sending agent may pre-weight outgoing messages in one approach, while the other approach may weigh in received messages. It is important to note that an agent has a connection with itself. An agent takes an equal share of its contribution in a uniform mixing matrix when aggregating received contributions from neighbors. All columns of matrix W_m sum to one, making this matrix column stochastic. If all agents have equal numbers of in and out neighbors (which is not the case in W_m), the matrix may be doubly stochastic (rows and columns sum to one).

A few examples of the most common network topologies are shown in figure 4. Depending on the topology used, an arbitrary number of in-out neighbors can be set. Due to their design, ring, fully connected, or torus topologies do not allow setting a custom number of neighbors. An agent in ring topology has two neighbors in undirected communication and one neighbor in directed communication. There is no difference between undirected and directed communication for a fully connected graph. However, the number of neighbors is fixed to $N-1$ in a network of N fully connected nodes. Sparse networks have the ability for each node to choose the number of neighbors, and in directed communication, an agent may have more in-neighbors than out-neighbors or vice-versa.

B. Learning process

Each agent's goal is to train its local model by leveraging its local dataset D_i and information received from its neighbors to minimize its average loss function F_i. In an IID data environment, an agent's local dataset D_i can be assumed to be a uniform sample example in the global dataset D. At the same time, in a non-IID environment, this assumption is not valid. An agent uses this local data D_i to train a local model by calculating mini-batch gradient $\bigtriangledown F_i(x_i; \xi_i), \xi_i \sim D_i$ and updating its local model $x_i = x_i - \eta F_i(x_i; \xi_i), \xi_i \sim D_i$ for E batch iterations, where η denotes the learning rate. Parameter E is set to one when the communication is performed after each training batch or increased to any arbitrary number to increase local computation and reduce communication steps.

The local training step is followed by a communication step in which agents exchange information. Algorithm 1 shows an abstract agent training scheme. Agents initialize their respectable model parameters with identical values as it aids in a faster and smoother learning curve. The learning process is executed in parallel, i.e., each agent performs train and communication steps in a loop. Different approaches include different message content exchanged between agents, and commonly, an agent must send its message to all its out-neighbors and receive all messages from its in-neighbors. A loss of a message in this phase may stall the overall learning process since an agent may forever wait for a message that never arrives. This one stalled agent may trigger a domino stalling effect of other agents. The neighbors of the stalled would be the first ones to wait to receive a message, followed by the neighbors of neighbors and so forth. We term this synchronous undirected communication. If an approach has

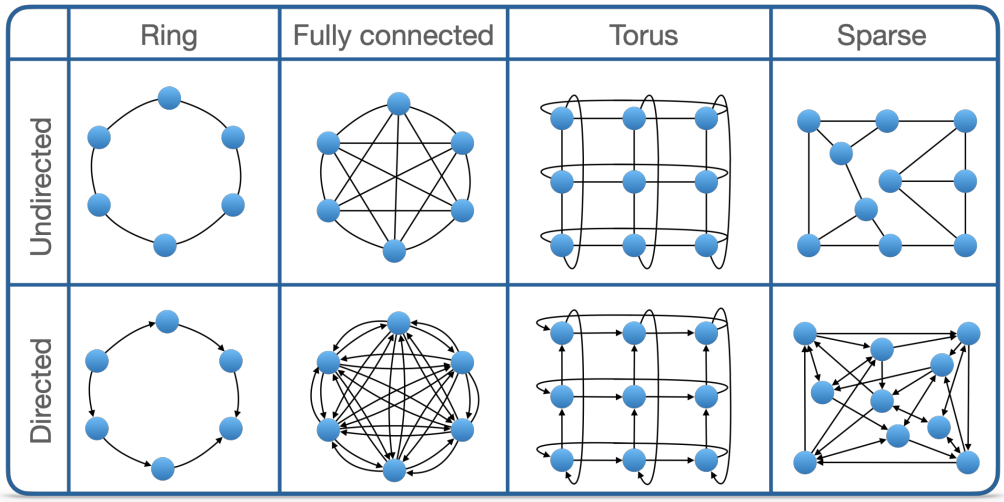

Fig. 4. Examples of network typologies in a undirected and directed communication.

a synchronization barrier but supports directed communication, we term it synchronous directed communication. Several approaches modify the communication step so that an agent continues the training process and processes messages upon arrival. We term this asynchronous directed communication. Another possible occurrence is when two agents communicate asynchronously (not disturbing the learning process) but require that messages are sent and received between agents. We term this case asynchronous undirected communication.

Algorithm 1 Agents abstract training process

Require:
Initialize $\eta > 0$, agents A, communication matrix W, number of local batch iterations E
x_i = 0 for all agents i $\in A$ ▷ Initialize all agents models to identical value

repeat for agent $i \in A$ ▷ In parallel
  for $e = 0, 1, 2, ..., E$, at agent, i **do**
    $\xi_i \sim D_i$ ▷ Sample new mini-batch from local distribution
    $x_i = x_i - \eta F_i(x_i; \xi_i)$ ▷ Train model on batch
  SEND($x_i W_{ji}$) ▷ Send model to out-neighbors
  RECEIVE($x_j W_{ij}$) ▷ Receive model from in-neighbors
  $x_i =$ AGGREGATE($x_j W_{ij}$)
until Maximum iteration reached

Output: $\frac{1}{N}\sum_{i=1}^{N} x_i$ or $x_i \forall i \in A$ ▷ Output is either one global model produced by averaging all models or personal model for each agent

C. Learning objective

The learning objective of collaborative learning is most commonly to produce a single global model by averaging the model parameters of all agents at the end of the training process:

$$x_{global} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

A neural model trained multiple times with different initial parameter values may find many different "optimal" solutions each run. The variance between solutions is reduced when model averaging is performed, often producing a robust model that performs fairly on the test dataset. Producing one global model is generally considered in big machine learning facilities to speed up the learning process by employing multiple training agents. Averaging model parameters might be troubling when specific layers are used. As an example, Batch Normalization layer (BN) [27] imposes averaging problems because it contains some statistical parameters that represent the statistical information, such as the mean and variance of the feature maps, which are solely contained in the normalization layers. BN is a normalization technique that is widely used in DNN [28–31] as it aids in smoother convergence and stability when training deep neural networks. Authors of [27] argue it reduces internal covariate shift, which dramatically accelerates the training of DNN models, makes training more resilient to large learning rates, and prevents model gradients from exploding or getting stuck in the poor local minima. The utilization of these favorable properties of the BN layer is somewhat limited in model averaging because special techniques must be developed to include statistical information produced by each BN layer in the aggregation phase. FedBN [30] and MTFL [31] use BN layers in FL but do not include them in the averaging process. Instead, BN layers stay only on the client, representing a form of model personalization. FedDNA [32] decouples standard model aggregation and statistical information aggregation by aggregating statistical parameters with an importance weighting method to reduce the divergence between the local models and the central model. Statistical parameters are optimized collaboratively by an adversarial learning algorithm based on a variational autoencoder (VAE).

Custom aggregation server-side rule is also considered in FedBS [33] as the mean of aggregating agents' statistical information.

Alternatively, each agent may have its learning objective, which means that the result of the training process may be one model per agent. Here, each model is personalized to each agent, performing best when used by the agent who trained the model and performing with deficient performance if used on an agent with different data distribution.

D. Approaches

Recently studied approaches are based on or modify the algorithm presented in 1. Table IV-D list these approaches and classifies some of their main properties.

Topology. A substantial number of approaches is open to using any topology [34, 36, 38, 39, 44–50, 54]. We classified these typologies based on the analysis conducted by the authors in their respectable papers. Approach working in any topology is a benefit since organizing complex typologies in the wild is not an easy task. Constraining the learning process to only some of the typologies [35, 40, 51–53] limits the applicability of the approach. Note that some approaches propose only communicating with one random node in the network [37, 41–43] which means that a node must have a list of all nodes in the network at all times. To conform to the algorithm 1, in these approaches, we can presume that the communication matrix W is a directed ring that changes each communication round. Some approaches [44, 45, 48] use this idea with neighbor communications. Even though these approaches support any topology, each agent only communicates with one random neighbor in each communication step. In theory, lowering the communication may negatively impact synchronization between agents. Column *# W_{ij}/step* of table IV-D displays the number of communicating neighbors in each each agent's communication step. The absence of a number implies that the number of neighbors depends on the chosen topology.

Communication direction. Since the topology often dictates the number of neighbors, it is important to analyze whether the communications are directed or undirected. In an example of undirected ring topology, each agent has two neighbors, while in the directed ring topology, each agent has only one neighbor (see figure 4). A majority of approaches rely upon or only support undirected communication, while some approaches [37, 38, 40, 43, 46] only work with directed communication. Only OverlapSGD [36] supports both directed and undirected communication.

Synchronisation. Synchronization is an important part of decentralized systems. It dictates a strategy that an agent obeys when communicating with neighbors. There is often a synchronization barrier in synchronous systems that stops the agent's learning process while all messages are exchanged. This approach can be as simple as sending messages to all out-neighbors and waiting for all in-neighbor messages. Given that all agents must perform the same steps, all agents can only continue the learning process once all messages are exchanged between neighbors. Since we study decentralized approaches, there is no central authority controlling the learning process. However, agents may follow a central clock for synchronization purposes. Alternatively, the *asynchronous time model* [55] enables each agent to locally track time with independent clock by ticking at a Poisson distribution [34, 46, 47]. Since all local clocks can be considered IID, they can equivalently be considered as a single global clock ticking at a rate N Poisson process, which wakes one or more network agents uniformly at random. This is especially useful when two agents must wake up and communicate simultaneously.

Combining the synchronization and communication properties, we can classify all the approaches to the groups mentioned above. Synchronous undirected communication [34–36, 39, 49, 50, 52–54] implies that all agents have a synchronization barrier in which all messages are exchanged in an undirected communication. We note that authors of [36] proposed an approach to prevent waiting at the synchronization barrier by potentially delaying synchronization for a specific number of iterations. Here, messages are processed as they come, but they still need to come in the expected order. The authors of [50] also proposed an asynchronous approach in which the agent firstly asks which neighbors will participate in the following communication step. We interpret this approach also as synchronous since there is no difference compare to the standard approach besides the potentially smaller number of neighbors. Querying alive neighbors could instead be classified as an attempt at a fault-tolerant or strangler-free solution. Communication can be executed in a directed manner but still require synchronization by requiring all neighbor in-messages before continuing the learning process [36, 38, 40]. In asynchronous undirected communication [34, 41, 42, 44, 45, 47, 48, 51], the communication between agents does not impede the learning process but still requires that both agents exchange messages between themselves in the same time interval. The most lenient of the four is the asynchronous directed communication [37, 43, 46]. Here, the communication is asynchronous and does not impede the learning process, nor does it require a reply message from the receiving agent. Each agent performs local training steps undisturbedly and only aggregates contributions received from in-neighbors as they come.

Objective, model and heterogeneity. Optimizing each agent's model performance (or loss function) is considered in [40, 42, 44–50], while other approaches form an aggregated global model at the end of the learning process. It is important to note that approaches [44–46, 48, 49] are restricted to linear models, and Dada [47] extend these works with a boosting-based approach. Other approaches are applicable to DNN models, while authors of [41] only offered a theoretical analysis without any application in mind. Out of the listed approaches, only the approach presented in [42] offers support for heterogeneous model architectures by splitting the network into slices and averaging each slice with the specified peer group. Each agent has complete freedom in defining which parts of the neural network are averaged. This is a positive feature of a decentralized system since agents may have different performance capabilities, requirements, and amounts

Approach	Topology	Comm.	# W_{ij}/step	Sync	Objective	Model	Heterogeneus
Colin *et. al* [34]	complete, ring, watts-strogatz	U	1	Sync, Async	global	NN	x
GossipGraD [35]	hypercube, dissemination-based	U	1	Sync	global	NN	x
OverlapSGD [36]	any	D, U	-	Sync	global	NN	x
GoSGD [37]	sent to random node	D	1	Async	global	NN	x
Online Push-Sum [38]	any	D	-	Sync	global	NN	x
CHOCO-SGD [39]	any	U	-	Sync	global	NN	x
Lalitha *et. al* [40]	aperiodic graph	D	-	Sync	individual	NN	x
Pilet *et. al* [41]	sent to random node	U	1	Async	global	-	x
Pilet *et. al* [42]	sent to random node	U	1	Async	individual	NN	✓
PeerSGD [43]	sent to random node	D	1	Async	global	NN	x
Vanhaesebrouck *et. al* [44]	any (sent to random neighbor)	U	1	Async	individual	Linear	x
DJAM [45]	any (sent to random neighbor)	U	1	Async	individual	Linear	x
Bellet *et. al* [46]	any	D	-	Async	individual	Linear	x
Dada [47]	any	U	-	Async	individual	Linear	x
CDPL [48]	any (sent to random neighbor)	U	1	Async	individual	Linear	x
Li *et. al* [49]	any	U	-	Sync	individual	Linear	x
DP-SGD [50]	any	U	-	Sync	individual	NN	x
AD-PSGD [51]	ring, ring-based ($2^i + 1$ hop neighbors)	U	1	Async	global	NN	x
D-PSGD [52]	ring	U	-	Sync	global	NN	x
D^2 [53]	ring	U	-	Sync	global	NN	x
Cooperative SGD [54]	any	U	-	Sync	global	NN	x

Table 1. Analysis and comparison of main decentralized properties between studied approaches. Topology column list all possible typologies studied in each respectable study. Column *Comm.* list if the communication is directed (D) or undirected (U). Column *# W_{ij}/step* shows the number of neighbors with whom an agent communicates in each communication round. If there is no number in the table, that means that an agent communicates with all neighbors prescribed by the communication topology. Some approaches only communicate with one neighbor, instead of respecting the communication topology (marked with number one). Column *Sync* describes whether there is a communication barrier to which an agent must obey, or the communication with the neighbors is asynchronous. The objective of the learning process may be one global (aggregated) model, or an individual model for each agent. This is described in the column *Objective*. Some of the approaches are strictly bounded to linear models, and are not applicable to NN models. Column *Model* states the model type used by each approach. Out of the listed approaches, only approach presented in [42] offers support for heterogeneous model architecture (column *Heterogeneous*).

of training data.

E. Communication compression

Communication between agents can present a challenge when the model architecture comprises millions of parameters. For example, the popular ResNet-50 architecture comprises over 23 million parameters and has 97MB in size. Communicating big models over the internet may take a long time which may slow down the learning process, especially the synchronous approaches. Strides were made in this area to reduce the size of the outgoing model parameters by applying *gradient compression*. The main approaches in compressing the models are gradient quantization and gradient sparsification. Gradient quantization [56] lowers the precision values representing model weights to a lower precision order. For example, rather than representing each weight as a float 64, weights are represented as a float8 (or even an integer). Reducing the precision of weight values also reduces overall model knowledge, which is the cost of reduced weight size. Gradient sparsification technique [57] limits the weights exchange if the absolute gradient values exceed a predetermined threshold. Only weights exceeding the threshold are allowed to be transmitted. Increasing the threshold reduces the number of exchanged weights; thus, communication is also reduced. Tuning this threshold parameter can be a challenge due to gradient value variations. Out of the approaches mentioned earlier, only CHOCO-SGD [39] considers using either gradient sparsification or gradient quantization. Experiments showed that this gradient compression method could reduce the total number of exchanged bits while reaching almost the same performance as the exact algorithm without communication restrictions.

F. Privacy

Data stored on agents is their personally generated data which is never shared with the outside world. However, when training a model, model parameters could learn this dataset very well, revealing sensitive information when transmitted to the neighboring peer. Differential privacy [58–60] offers strong mathematical guarantees for privacy by adding random noise to the model parameters during the training process. This way, the models do not learn or remember any specific details about any agent's data. The noise is, in general, generated using a Gaussian distribution [61]. The gradient clipping technique is commonly performed alongside adding random noise. Gradient clipping bounds the influence of each individual training example to the gradients by clipping each gradient in ℓ_2 norm; a gradient vector g is replaced by $g/max(1, \frac{\|g\|_2}{C})$ for a clipping threshold C.

Two metrics are used to express the privacy guarantee for a DNN model:

- Delta (δ) - bounds the probability of the privacy guarantee not being respected. In general, it is recommended to set this parameter to the inverse value of the dataset size. For a dataset of 10,000 examples, the value should be set to 10^{-5}.
- Epsilon (ε) - a measure of the privacy budget. A smaller ε value implies a better privacy guarantee. This parameter is a bound of a variation of particular model output. Since ε is an upper bound, a larger value could still mean a good privacy guarantee in practice.

To formalize, let M be a randomized mechanism taking a dataset as an input, and let $\varepsilon > 0, \delta \geq 0$, then M is (ε, δ)-differentially private if for all agents datasets $D = \{d_1, ..., d_i, ..., d_n\}, S' = \{d_1, ..., d'_i, ..., d_n\}$ differing in a single data point and for all sets of possible outputs $O \subseteq range(M)$ holds that:

$$Pr[M(S) \in O] \leq e^{\varepsilon} Pr[M(S') \in O] + \delta$$

In a decentralized system, a (ε, δ)-differential privacy guarantees that for any neighboring agents datasets D_i and D_j, it is very likely that retroactively, the value of $M(D_i)$ will have a similar probability to be generated from the D_i or D_j dataset. For all agents, the absolute value of the privacy loss will be bounded by ε with a probability of privacy guarantee of at least $1 - \delta$. Differential privacy is extensively studied in Federated Learning [62–65] but requires more research in decentralized systems. In [46] differential privacy was obtained by generating noise from a Laplace distribution, while the authors of [41] only provided a theoretical analysis of an approach where agents only add random noise for a specific number of communications. Authors of [50] had a similar idea. Their approach was to lower the generated noise scale over time to ensure better model accuracy. A visual example of the privacy budget impact on the data reconstruction is shown in figure 5 [66]. Figure provides four visual examples on four datasets: MNIST [3], Fashion-MNIST [67], LFW [68], and CIFAR10 [69] under three scenarios: (I) non-private, (II) DP-baseline with fixed parameter setting of clipping bound $C = 1$ and noise scale $\sigma = 1$, and (III) DP-baseline with clipping bound $C = 4$ and noise scale $\sigma = 6$, where σ is standard deviation of a Gaussian noise function, and C is the clipping threshold of the ℓ_2 norm of gradient vector. An example of different privacy budgets is given in the example of MNIST reconstructions with gradients clipping of a maximum ℓ_2 norm of 1 and $\varepsilon = 10^{-5}$ [70] (see figure 6). It is important to note that by increasing the privacy of the amount of added noise and gradient clipping, the overall training loss also increases [70]. This is the reason why limiting the number of iterations in which the differential privacy methods are applied was initially proposed [41, 53].

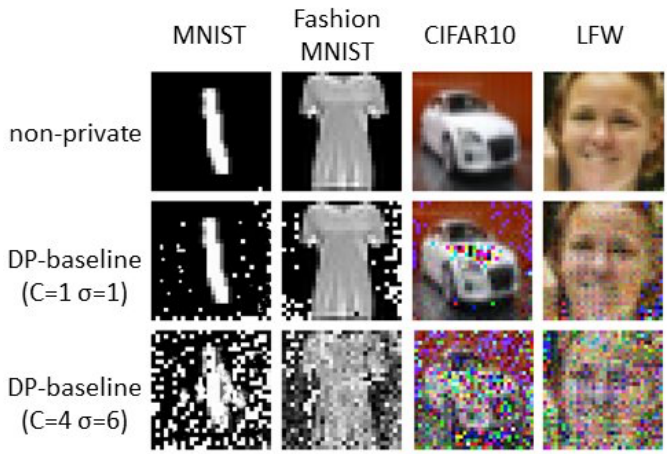

Fig. 5. Visual examples on MNIST, Fashion-MNIST, LFW, and CIFAR10 datasets under three scenarios: non-private, DP-baseline with fixed parameter setting of clipping bound $C = 1$ and noise scale $\sigma = 1$ and DP-baseline with clipping bound $C = 4$ and noise scale $\sigma = 6$.

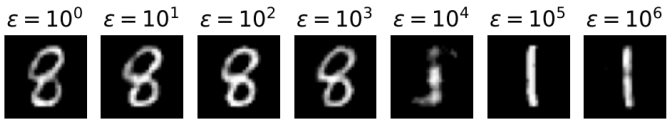

Fig. 6. Example of MNIST reconstructions under different differential privacy budgets, with gradients clipping of a maximum ℓ_2 norm of 1 and $\varepsilon = 10^{-5}$.

G. IID vs. Non-IID data

The agent's local data should never be exchanged with the neighbors to ensure privacy. However, approaches such as [34, 35] consider shuffling the data between agents to aid in convergence and accuracy, and only IID data environment is considered or assumed in [36, 37, 43–46, 48–52, 54]. An updated version of GoSGD [43] performed well in a churn setting (agents joining/leaving the network) in an IID data environment. However, the proposed approach could not converge in a non-IID data environment where each agent received examples of only one class. Authors of CDPL [48] also observed that in the case of peers holding non-iid data, CDPL failed to drive the peers to generalize their models and reach a good accuracy, even after running the algorithm for 200 rounds. Efforts were made to address the non-IID data environment. In [38], the authors split the dataset into two subsets: the stochastic data and the adversarial data. The stochastic data was generated by allocating a fraction of samples (e.g., 50% of the whole dataset) to agents randomly and uniformly. The adversarial data was generated by random sampling on the remaining dataset to produce N clusters and then allocating every cluster to an agent. Experiments showed that decreasing the stochastic-adversarial ratio causes increase in the average loss value. A non-IID data environment was considered in [39] where each agent only received samples from one or two classes. The authors reported that the IID data experiments performed better than the non-IID data experiments. However, it is unclear by which amount since they did not quantify the results. Similar data separation was considered in [53]. In a five agent setting, and by assigning the agents with only the samples of two classes, the authors show that D^2 outperforms D-PSGD [52] under high data variance, achieving comparable performance as a centrally trained model. Comparing the IID and non-IID data environments for their approach, the authors of [40] reported an accuracy drop of around 2% on the MNIST fashion dataset [67] in a two agent network. In [42] the authors experimented with the FEMNIST [71] dataset of 62 classes of handwritten characters by assigning samples from exactly one writer to each agent. All writers' sets of samples were limited to the same size to eliminate bias in results. Since the authors were conducting experiments measuring accuracy depending on the averaging level and the number of mini-batches, no details regarding the class distribution between

agents were provided, and no IID or centrally trained baselines were provided. For the Dada [47] approach, experiments were conducted on realistic datasets that are naturally collected at the user level, such as Human Activity Recognition With Smartphones (Harws) [72] and Vehicle Sensor [73]. Results show that the proposed method achieves a small increase in accuracy when agents train their models collaboratively, compared to individual training without any communication. In a distributed network of sensors, a multi-task approach was also considered [49]. Experiments were conducted on temperature sensors of outside and in-house temperatures, where each sensor was considered a single task. Results, however, only demonstrate a minute mean error loss when using the collaborative approach proposed in the study using regression models.

V. Experiments on IID vs non-IID data

To investigate the differences in convergence and top accuracy, we evaluated several methods in the IID and non-IID data settings using the ring and sparse topologies. For each experiment, three separate runs were conducted for each approach and averaged to produce the final result.

Baseline training methods. We evaluated several methods: (I) D^2 [53] as a representative of the parallel stochastic gradient descent family (eg. [50–52]) in which the optimal topology is undirected ring graph (doubly stochastic symmetric); (II) SGP [74] from the stochastic gradient push family (eg. [36, 38]); (III) GoSGD [37] from decentralized gossip exchange approaches (eg. [35]). Approaches D^2 and SGP are simulated by synchronously training all agents using a *uniformly mixing communication matrix*. A uniformly mixed communication matrix implies that all received models from peers are equally important. GoSGD is also simulated by synchronously training all agents. We modified the communication behavior of the GoSGD approach so that the agent chooses its neighbor(s) based on the tested topology. The probability of sending a model to each peer is set to 0.1 in all experiments. In all compared methods, agents first perform a local training step, followed by a communication step, repeating these two steps in a loop (see figure 7).

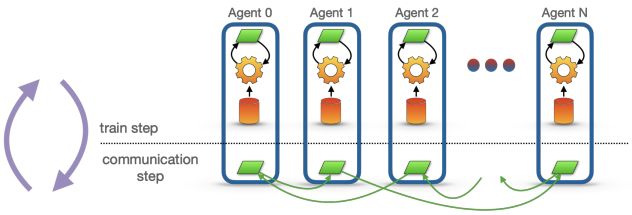

Fig. 7. Synchronous agent training, in which all agents first perform a local training step, followed by a communication step, repeating these two steps in a loop.

A. Methodology

Dataset. We evaluated the approaches on the MNIST [75] and Reddit datasets [76]. The MNIST [75] dataset consists of handwritten digits images and is used in a classification task. The MNIST dataset was evaluated in multiple settings: 1) IID data setting [9] that distributes all classes uniformly across different agents; 2) *pathological* non-IID data setting [9] that partitions the dataset so that each agent gets only two classes (out of 10); 3) *practical* non-IID data setting [22] that partitions the data between agents so that every agent has data from all the classes, but with different distributions; some classes have a higher probability than other. The Reddit dataset [76] consists of 1,660,820 unique Reddit users and their comments, and each comment can have multiple sentences. In the experiments, a single agent dataset consisted of all comments from a unique Reddit user and is used for solving a next-word prediction task. Furthermore, agent datasets are divided into training, validation, and test subsets in a 60%-20%-20% split. Following the experiments from previous work [16, 77–80], the vocabulary size was set to 10,000 most common words. Other words are characterized as an out-of-vocabulary token and are not considered when calculating prediction accuracy.

Agent model description. Model architecture used in [9] was used for the MNIST classification task. For the next-word prediction task, the model architecture applied on agents is adopted from [79, 80], however, the Long Short-Term Memory (LSTM) [81] layer was replaced with a recurrent neural network layer called Gated Recurrent Unit (GRU) [82].

Metrics. Average User model Accuracy (UA) [31] metric was used to measure the overall learning process performance. UA measures the average accuracy across all agents on their local test data, and can be expressed as:

$$\text{UA} = \frac{1}{n}\sum_{i=1}^{n} acc_i$$

where acc_i is the prediction accuracy of model i on local test dataset i (both owned by agent i). Prediction accuracy is calculated as the fraction of correct predictions (TP and TN) over all predictions (TP, TN, FP, and FN):

$$acc_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}.$$

B. Experiments

For the MNIST experiments, the batch size was set to 32, and the learning rate to 10^{-3}. In the IID MNIST data setting, data was distributed uniformly across agents. In the *pathological* non-IID data setting, each agent only received examples of two classes [9]. For the *practical* non-IID data setting, each agent received 80% of the two dominant classes, and 20% of all other classes [22]. For the Reddit experiments, the batch size was set to 50, and the learning rate to 5×10^{-3}. All experiments start with identical agent model parameters and identical agent peer connections.

Experiments in ring topology. We simulated the P2P training process of 100 agents on D^2, GoSGD, and SGP approaches in a directed and undirected ring topology for 100 agent

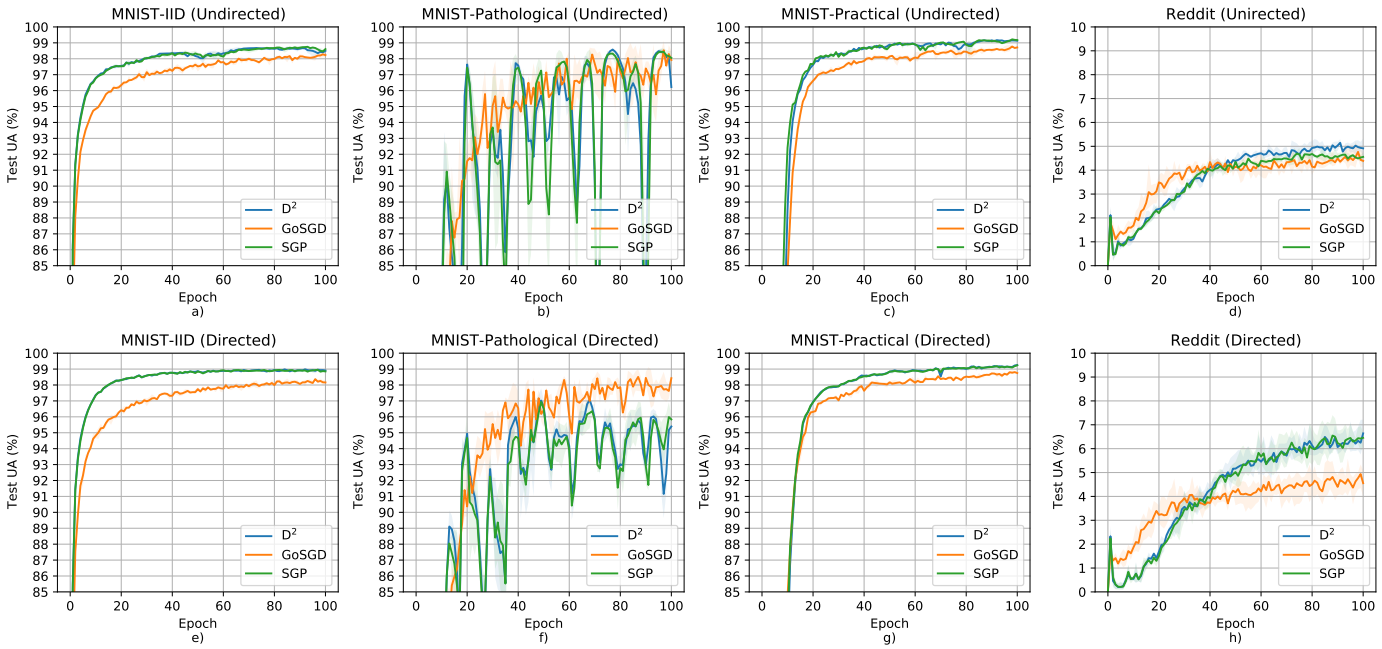

Fig. 8. Test UA of 100 agents for D^2, GoSGD, and SGD in undirected/directed ring topology using the IID MNIST, *pathological* non-IID MNIST, *practical* non-IID MNIST and Reddit datasets.

epochs. Figure 8 shows the results of the simulations. In general, directed communication provided smoother and less variable convergence curve for the ring topology. IID MNIST is a slightly easier task to learn compared to the *practical* MNIST, resulting in faster UA convergence. In the case of the *pathological* MNIST dataset, there are significant deviations in accuracy, especially in undirected communication. For the Reddit dataset, GoSGD struggles both in undirected and directed ring topology, while D^2 and SGP achieve greater top AU in directed communication. There is a noticeable difference between accuracies achieved for the MNIST and Reddit dataset. The Reddit dataset can also be considered as a classification task with 10,000 classes, which makes the task very difficult (max test UA is around 7%).

Experiments in sparse topology. D^2, GoSGD, and SGP approaches were also evaluated in undirected/directed sparse topology with fixed number of three in-out neighbors. Figure 9 shows the results of the simulations in sparse topology. As in ring topology, IID and *practical* MNIST are relatively easy tasks to learn, while *pathological* MNIST caused even worse UA deviations in sparse topology. For the Reddit dataset, GoSGD performed fairly even in directed and undirected communication, but achieved better results compared to the ring topology. In contrast, D^2 and SGP performed worse in both directed and undirected communication compared to the ring topology.

Centrally trained model on pooled data. When training one single model on all Reddit training data pooled, the model, on average, achieved a maximum accuracy of 12.8%. For the MNIST dataset, maximum accuracy that a single model achieved was 99.05%. In both ring and sparse topologies, D^2 and SGP achieved a top UA accuracy of around 99% for the IID and *practical* MNIST but have had difficulties converging using the *pathological* dataset, especially in sparse topology. The most noticeable accuracy difference between centrally and peer-to-peer trained models is for the Reddit dataset. Centrally trained model achieved around 40% higher relative mean accuracy. This higher accuracy discrepancy between MNIST and Reddit dataset is probably due to the way MNIST dataset were partitioned. In all three MNIST settings, agents had received identical data distributions of training and testing data, while the Reddit dataset contains data examples that are only present in the training or testing data. Moreover, Reddit dataset was used for a next-word prediction task with a thousandfold more output classes.

VI. Open problems and future research

There is undoubtedly a need for more research on decentralized learning techniques and the communication process between the agents. Research is especially important in volatile peer-to-peer networks in which agents constantly join and leave the network, commonly referred to as churn. Authors of PeerSGD [43] showed that their variant of the GoSGD [37] approach performed well under high network churn, but only in strong IID data setting. As mentioned before, we classified the communication of PeerSGD and GoSGD as asynchronous directed communication, where each agent sends a message to its neighbor(s) asynchronously and without waiting for a response. Under network churn, this feature means that an agent would not be affected by a neighbor dropping out of the network. Sending agent is not affected if the receiving agent is not online. It may only affect the receiving agent's learning process since it did not receive an update from the sending agent. On the other side of the spectrum are synchronous undirected and directed communication approaches.

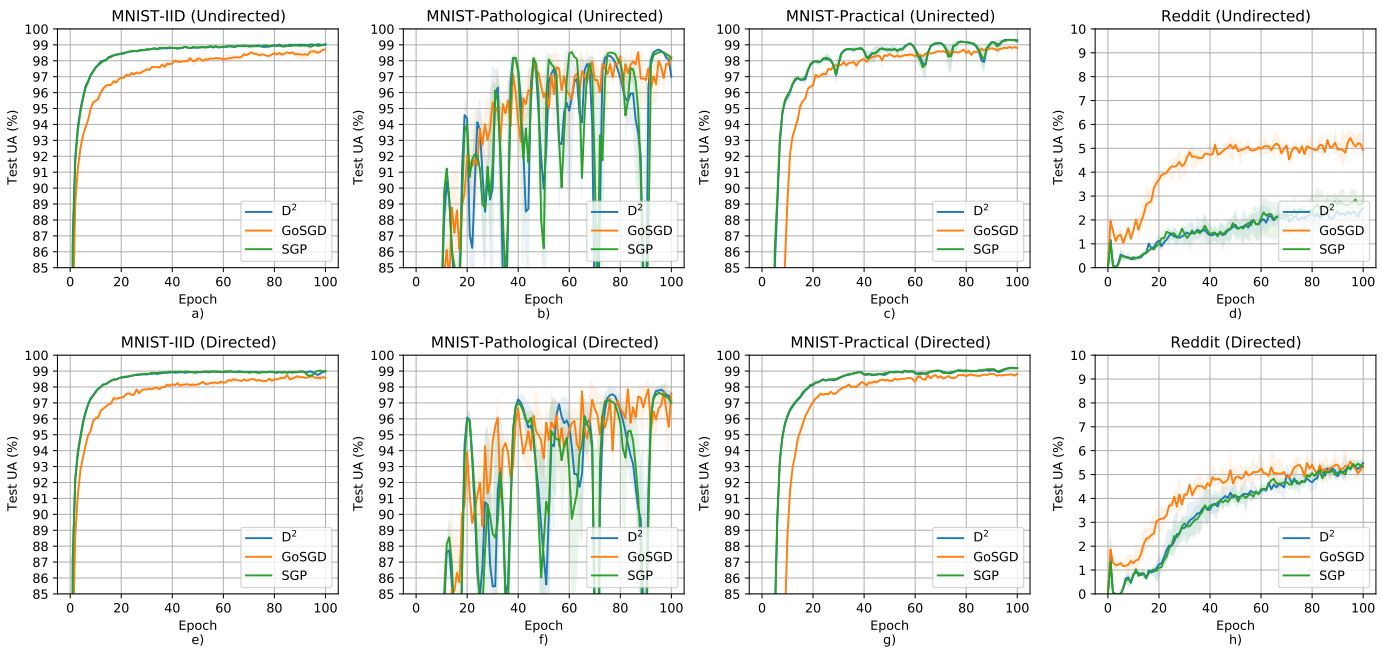

Fig. 9. Test UA of 100 agents for D^2, GoSGD, and SGD in undirected/directed sparse topology using the IID MNIST, *pathological* non-IID MNIST, *practical* non-IID MNIST and Reddit datasets.

An absence of an agent in synchronous communication may halt the learning process of all involved agents. If an agent is absent, its neighbors will wait for a response indefinitely, causing the neighbors of the neighbors to wait for a response in the next round, triggering a halting domino effect. Halting may also affect agents in asynchronous undirected communication but on a smaller scale. The two agents exchanging information must send and receive messages before continuing their learning process. If one of the agents does not send their message, the other may also wait indefinitely for a response.

There is no supervision over the learning process in decentralized systems as in FL. This absence of control imposes attack possibilities that other agents may explore to impede a learning process of an agent. Of course, even FL is not immune to malicious attacks (refer to [11, 83] for more details). Online Push-Sum [38] considered a single-sided trust weighted communication matrix instead of a uniformly mixed matrix. The matrix weights represent the trust between the nodes, i.e., an agent's impact on another agent. The matrix was presumed in the research and not learned by the agents. PeerSGD [43] analyzes the model upon receiving it. The receiving agent calculates the loss for the receiving and local model on a data batch. The receiving model will be rejected if the absolute difference between losses is greater than a preset threshold. A threshold parameter is considered in CDPL [48] to avoid a potential accuracy decrease in the agent's model by rejecting the received model if its accuracy on the test dataset is not acceptable. Evaluating model performance on local data may help an agent preserve itself from malicious attacks, but this is still an area of open research. A malicious attack may also be considered as changing the model's behavior when predicting just one of the classes. The model could still perform well on all other classes and "bypass" the accuracy or loss checks. For example, in autonomous driving, falsely predicting just one class may have serious consequences.

Commonly, the communication matrix is predetermined upfront. A predetermined matrix is a vital presumption, especially in multi-task approaches. A multi-task approach Dada [47] considered the process of relations forming between agents. In Dada, the peer-to-peer connections are learned along with the model based on agents' tasks. For the experiment using a synthetic Moon dataset, agents were clustered based on the moon rotation. The assumption is that clustering agents with similar learning goals will enhance their learning ability. Whether there might be some benefits in mixing agents between clusters is an interesting future research question. Since connections between agents can be learned through services such as random peer sampling service (RPS) [84], and potential neighbors can join and leave at any time, it is crucial to study the potential sparsity of the network. Current research is mainly focused on a balanced and fixed number of neighbors. However, an agent might be limited to communication with fewer neighbors than other agents. Reasons may be that peers reject communication requests due to already reaching the neighbor limit, or, in a multi-task setting, peers reject communication due to differences in objectives.

Energy consumption is a concern especially highlighted on battery-powered devices such as mobile devices. An economic research direction should investigate the trade-offs between communication and local computation. This research should investigate the effects of increasing the amount of communication and decreasing the amount of local computation, or vice-versa. By knowing the cost of communication and local

computation for each agent, an optimal strategy would be applied to each agent.

With a vast number of different devices present, such as sensors, mobile devices, or laptops, there is a need for a collaborative approach that supports heterogeneous models. By slicing the network into pieces, in a multi-task environment, agents were able to average only parts of the network in [42]. This approach still requires that two models share the architecture of some layers. Future research could consider the model distillation approach as a method of knowledge transfer between heterogeneous models.

VII. Conclusion

Peer-to-peer learning is an area of great research interest. This paper presents an overview of the recent advances and general approaches in decentralized learning techniques. The main difference between *parameter-server* (FL) and decentralized learning approaches is the process of orchestrating the collaboration between agents. The lack of central authority guiding the collaboration in decentralized systems puts more accountability on agents, making them responsible for orchestrating the collaboration process. Before collaborating, agents must discover each other. Since the collaboration graph is commonly assumed in studies, future research needs to study agent discovery and conditions of forming connections between agents. When collaborating, it is essential to consider what information is being conveyed and how. Agent's privacy should be preserved to prevent private data leakage, and the communication between agents must be fault-tolerant to prevent errors. Directed asynchronous communication presents a good solution, but more research needs to be conducted to understand churn and network sparsity scenarios. In decentralized unsupervised systems, invalid models may be exchanged between agents maliciously or unknowingly. Future studies should address this problem in more detail rather than just validating the model on test data. To better understand the applicability of these decentralized approaches in real systems, studies should be carried out with realistic non-IID data. As considered agents are mostly battery-powered, particular interest should be dedicated to optimizing the amount of local computation and communication. Future research should also address model heterogeneity. An approach supporting knowledge transfer between heterogeneous models would enable each agent to own a custom personalized model that best meets its needs.

References

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[3] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[4] Xiang Li et al. "On the Convergence of FedAvg on Non-IID Data". In: *International Conference on Learning Representations*. 2020.

[5] Kevin Hsieh et al. *The Non-IID Data Quagmire of Decentralized Machine Learning*. Sept. 2019. URL: https://arxiv.org/abs/1910.00189.

[6] Rong Yu and Peichun Li. "Toward Resource-Efficient Federated Learning in Mobile Edge Computing". In: *IEEE Network* 35.1 (2021), pp. 148–155. DOI: 10.1109/MNET.011.2000295.

[7] Hui Lv, Si Shi, and Dogan Gursoy. "A look back and a leap forward: a review and synthesis of big data and artificial intelligence literature in hospitality and tourism". In: *Journal of Hospitality Marketing & Management* 31.2 (2022), pp. 145–175. DOI: 10.1080/19368623.2021.1937434. eprint: https://doi.org/10.1080/19368623.2021.1937434. URL: https://doi.org/10.1080/19368623.2021.1937434.

[8] Karan Aggarwal et al. "Has the Future Started? The Current Growth of Artificial Intelligence, Machine Learning, and Deep Learning". In: *Iraqi Journal For Computer Science and Mathematics* 3.1 (Jan. 2022), pp. 115–123. DOI: 10.52866/ijcsm.2022.01.01.013. URL: https://journal.esj.edu.iq/index.php/IJCM/article/view/100.

[9] Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, Apr. 2017, pp. 1273–1282. URL: https://proceedings.mlr.press/v54/mcmahan17a.html.

[10] Daliang Li and Junpu Wang. "FedMD: Heterogenous Federated Learning via Model Distillation". In: *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with NeurIPS* (2019).

[11] Hongyan Chang et al. *Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer*. 2019. URL: http://arxiv.org/abs/1912.11279.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *NIPS Deep Learning and Representation Learning Workshop*. 2015. URL: http://arxiv.org/abs/1503.02531.

[13] Jianping Gou et al. "Knowledge Distillation: A Survey". In: *International Journal of Computer Vision* 129.5 (2021), pp. 1789–1819. ISSN: 1573-1405. DOI: 10.1007/s11263-021-01453-z. URL: https://doi.org/10.1007/s11263-021-01453-z.

[14] Ilias Diakonikolas et al. "Being Robust (in High Dimensions) Can Be Practical". In: *Proceedings of the 34th International Conference on Machine Learning - Vol-*

ume 70. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 999–1008.

[15] Peter Kairouz et al. "Advances and Open Problems in Federated Learning". In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210. ISSN: 1935-8237. DOI: 10.1561/2200000083. URL: http://dx.doi.org/10.1561/2200000083.

[16] Kangkang Wang et al. "Federated Evaluation of On-device Personalization". In: *ArXiv* abs/1910.10252 (2019). URL: http://arxiv.org/abs/1910.10252.

[17] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. *Adaptive Personalized Federated Learning*. 2021.

[18] Virginia Smith et al. "Federated Multi-Task Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 4425–4435. URL: https://proceedings.neurips.cc/paper/2017/file/6211080fa89981f66b1a0c9d55c61d0f-Paper.pdf.

[19] Mohammad Rostami et al. "Multi-Agent Distributed Lifelong Learning for Collective Knowledge Acquisition". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '18. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 712–720.

[20] Javad Mohammadi and Soheil Kolouri. "Collaborative Learning Through Shared Collective Knowledge and Local Expertise". In: *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2019, pp. 1–6. DOI: 10.1109/MLSP.2019.8918888.

[21] Roula Nassif et al. "Multitask Learning Over Graphs: An Approach for Distributed, Streaming Machine Learning". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 14–25. DOI: 10.1109/MSP.2020.2966273.

[22] Yutao Huang et al. "Personalized Cross-Silo Federated Learning on Non-IID Data". In: *CoRR* abs/2007.03797 (2020). URL: https://arxiv.org/abs/2007.03797.

[23] Rui Hu et al. "Privacy-Preserving Personalized Federated Learning". In: *ICC 2020 - 2020 IEEE International Conference on Communications ICC*. 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9149207.

[24] Manoj Ghuhan Arivazhagan et al. "Federated Learning with Personalization Layers". In: *ArXiv* abs/1912.00818 (2019).

[25] Yihan Jiang et al. *Improving Federated Learning Personalization via Model Agnostic Meta Learning*. 2020. URL: http://arxiv.org/abs/1909.12488.

[26] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. "Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 3557–3568. URL: https://proceedings.neurips.cc/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf.

[27] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Vol. 37. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[28] Pramod Kaushik Mudrakarta et al. "K For The Price Of 1: Parameter Efficient Multi-task And Transfer Learning". In: *ArXiv* abs/1810.10703 (2019), pp. 1–15. URL: http://arxiv.org/abs/1810.10703v2.

[29] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. "Asynchronous Federated Optimization". In: *ArXiv* abs/1903.03934 (2019). URL: http://arxiv.org/abs/1903.03934.

[30] Xiaoxiao Li et al. "FedBN: Federated Learning on Non-IID Features via Local Batch Normalization". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[31] J. Mills, J. Hu, and G. Min. "Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing". In: *IEEE Transactions on Parallel and Distributed Systems* 33.03 (Mar. 2022), pp. 630–641. ISSN: 1558-2183. DOI: 10.1109/TPDS.2021.3098467.

[32] Jian-Hui Duan, Wenzhong Li, and Sanglu Lu. "FedDNA: Federated Learning with Decoupled Normalization-Layer Aggregation for Non-IID Data". In: *Machine Learning and Knowledge Discovery in Databases. Research Track*. Ed. by Nuria Oliver et al. Cham: Springer International Publishing, 2021, pp. 722–737. ISBN: 978-3-030-86486-6.

[33] Meryem Idrissi, Ismail Berrada, and Guevara Noubir. "FEDBS: Learning on Non-IID Data in Federated Learning using Batch Normalization". In: *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2021, pp. 861–867. DOI: 10.1109/ICTAI52525.2021.00138. URL: https://doi.ieeecomputersociety.org/10.1109/ICTAI52525.2021.00138.

[34] Igor Colin et al. "Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1388–1396. URL: https://proceedings.mlr.press/v48/colin16.html.

[35] Jeff A. Daily et al. "GossipGraD: Scalable Deep Learning using Gossip Communication based Asynchronous Gradient Descent". In: *ArXiv* abs/1803.05880 (2018).

[36] Mahmoud Assran et al. "Stochastic Gradient Push for Distributed Deep Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR,

June 2019, pp. 344–353. URL: https://proceedings.mlr.press/v97/assran19a.html.

[37] Michael Blot et al. "Distributed optimization for deep learning with gossip exchange". In: *Neurocomputing* 330 (2019), pp. 287–296. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0925231218313195.

[38] Chaoyang He et al. "Central Server Free Federated Learning over Single-sided Trust Social Networks". In: *ArXiv* abs/1910.04956 (2019). URL: http://arxiv.org/abs/1910.04956.

[39] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. "Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 3478–3487. URL: https://proceedings.mlr.press/v97/koloskova19a.html.

[40] Anusha Lalitha et al. "Peer-to-peer Federated Learning on Graphs". In: *ArXiv* abs/1901.11173 (2019). URL: http://arxiv.org/abs/1901.11173.

[41] Amaury Bouchra Pilet, Davide Frey, and François Taïani. "Robust Privacy-Preserving Gossip Averaging". In: *SSS 2019 - 21st International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Pisa, Italy: Springer, Oct. 2019, pp. 38–52. DOI: 10.1007/978-3-030-34992-9_4. URL: https://hal.archives-ouvertes.fr/hal-02373353.

[42] Amaury Bouchra Pilet, Davide Frey, and François Taïani. "Simple, Efficient and Convenient Decentralized Multi-Task Learning for Neural Networks". In: *IDA 2021 - 19th Symposium on Intelligent Data Analysis*. Vol. 12695. Lecture Notes in Computer Science. Porto, Portugal: Springer, Apr. 2021. DOI: 10.1007/978-3-030-74251-5_4. URL: https://hal.archives-ouvertes.fr/hal-02373338.

[43] Robert Šajina, Nikola Tanković, and Darko Etinger. "Decentralized trustless gossip training of deep neural networks". In: *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. 2020, pp. 1080–1084. DOI: 10.23919/MIPRO48935.2020.9245248.

[44] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. "Decentralized Collaborative Learning of Personalized Models over Networks". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, Apr. 2017, pp. 509–517. URL: https://proceedings.mlr.press/v54/vanhaesebrouck17a.html.

[45] Inês Almeida and João Xavier. "DJAM: Distributed Jacobi Asynchronous Method for Learning Personal Models". In: *IEEE Signal Processing Letters* 25.9 (2018), pp. 1389–1392. DOI: 10.1109/LSP.2018.2859596.

[46] Aurélien Bellet et al. "Personalized and Private Peer-to-Peer Machine Learning". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, Apr. 2018, pp. 473–481. URL: https://proceedings.mlr.press/v84/bellet18a.html.

[47] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. "Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pp. 864–874. URL: https://proceedings.mlr.press/v108/zantedeschi20a.html.

[48] Karim Boubouh et al. "Robust P2P Personalized Learning". In: *2020 International Symposium on Reliable Distributed Systems (SRDS)*. 2020, pp. 299–308. DOI: 10.1109/SRDS51746.2020.00037.

[49] Jiyi Li et al. "Distributed Multi-task Learning for Sensor Network". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Michelangelo Ceci et al. Cham: Springer International Publishing, 2017, pp. 657–672. ISBN: 978-3-319-71246-8.

[50] Shangwei Guo et al. *Differentially Private Decentralized Learning*. June 2020. URL: http://arxiv.org/abs/2006.07817.

[51] Xiangru Lian et al. "Asynchronous Decentralized Parallel Stochastic Gradient Descent". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 3043–3052. URL: https://proceedings.mlr.press/v80/lian18a.html.

[52] Xiangru Lian et al. "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/f75526659f31040afeb61cb7133e4e6d-Paper.pdf.

[53] Hanlin Tang et al. "D^2: Decentralized Training over Decentralized Data". In: *Proceedings of the 35th International Conference on Machine Learning* 80.1 (2018), pp. 4848–4856. URL: http://proceedings.mlr.press/v80/tang18a.html.

[54] Jianyu Wang and Gauri Joshi. "Cooperative SGD: A Unified Framework for the Design and Analysis of Local-Update SGD Algorithms". In: *Journal of Machine Learning Research* 22.213 (2021), pp. 1–50. URL: http://jmlr.org/papers/v22/20-147.html.

[55] S. Boyd et al. "Randomized gossip algorithms". In: *IEEE Transactions on Information Theory* 52.6 (2006), pp. 2508–2530. DOI: 10.1109/TIT.2006.874516.

[56] Hanlin Tang et al. "Communication Compression for Decentralized Training". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et

al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/44feb0096faa8326192570788b38c1d1-Paper.pdf.

[57] Nikko Strom. "Scalable distributed DNN training using commodity GPU cloud computing". In: *Proc. Interspeech 2015*. 2015, pp. 1488–1492. DOI: 10.21437/Interspeech.2015-354.

[58] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284. ISBN: 978-3-540-32732-5.

[59] Cynthia Dwork. "A Firm Foundation for Private Data Analysis". In: *Commun. ACM* 54.1 (Jan. 2011), pp. 86–95. ISSN: 0001-0782. DOI: 10.1145/1866739.1866758. URL: https://doi.org/10.1145/1866739.1866758.

[60] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3–4 (Aug. 2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[61] Martin Abadi et al. "Deep Learning with Differential Privacy". In: CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 308–318. ISBN: 9781450341394. DOI: 10.1145/2976749.2978318. URL: https://doi.org/10.1145/2976749.2978318.

[62] Stacey Truex et al. "LDP-Fed: Federated Learning with Local Differential Privacy". In: EdgeSys '20. Heraklion, Greece: Association for Computing Machinery, 2020, pp. 61–66. ISBN: 9781450371322. DOI: 10.1145/3378679.3394533. URL: https://doi.org/10.1145/3378679.3394533.

[63] Antonious Girgis et al. "Shuffled Model of Differential Privacy in Federated Learning". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, Apr. 2021, pp. 2521–2529. URL: https://proceedings.mlr.press/v130/girgis21a.html.

[64] Zehui Zhang et al. "Privacy-enhanced momentum federated learning via differential privacy and chaotic system in industrial Cyber–Physical systems". In: *ISA Transactions* (2021). ISSN: 0019-0578. DOI: https://doi.org/10.1016/j.isatra.2021.09.007. URL: https://www.sciencedirect.com/science/article/pii/S0019057821004912.

[65] Yang Zhao et al. "Local Differential Privacy-Based Federated Learning for Internet of Things". In: *IEEE Internet of Things Journal* 8.11 (2021), pp. 8836–8853. DOI: 10.1109/JIOT.2020.3037194.

[66] Wenqi Wei and Ling Liu. *Gradient Leakage Attack Resilient Deep Learning*. preprint on webpage at https://www.researchgate.net/publication/357365445_Gradient_Leakage_Attack_Resilient_Deep_Learning. Dec. 2021.

[67] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. cite arxiv:1708.07747Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/. 2017. URL: http://arxiv.org/abs/1708.07747.

[68] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Oct. 2007.

[69] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "The CIFAR-10 dataset". In: *online: http://www.cs.toronto.edu/kriz/cifar.html* 55.5 (2014).

[70] Borja Balle, Giovanni Cherubin, and Jamie Hayes. "Reconstructing Training Data with Informed Adversaries". In: *2022 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2022, pp. 1556–1556. DOI: 10.1109/SP46214.2022.00127. URL: https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00127.

[71] Sebastian Caldas et al. *LEAF: A Benchmark for Federated Settings*. 2018. DOI: 10.48550/ARXIV.1812.01097. URL: https://arxiv.org/abs/1812.01097.

[72] D. Anguita et al. "A public domain dataset for human activity recognition using smartphones". In: *Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2013, pp. 437–442. ISBN: 978-2-87419-081-0. URL: http://hdl.handle.net/2117/20897.

[73] Marco F Duarte and Yu Hen Hu. "Vehicle classification in distributed sensor networks". In: *Journal of Parallel and Distributed Computing* 64.7 (2004). Computing and Communication in Distributed Sensor Networks, pp. 826–838. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2004.03.020. URL: https://www.sciencedirect.com/science/article/pii/S0743731504000255.

[74] Angelia Nedić and Alex Olshevsky. "Stochastic Gradient-Push for Strongly Convex Functions on Time-Varying Directed Graphs". In: *IEEE Transactions on Automatic Control* 61.12 (2016), pp. 3936–3947. DOI: 10.1109/TAC.2016.2529285.

[75] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[76] Sebastian Caldas et al. "LEAF: A Benchmark for Federated Settings". In: *ArXiv* abs/1812.01097 (2018).

[77] Andrew Hard et al. *Federated Learning for Mobile Keyboard Prediction*. 2018. URL: https://arxiv.org/abs/1811.03604.

[78] Sai Praneeth Karimireddy et al. "SCAFFOLD: Stochastic Controlled Averaging for Federated Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 5132–5143. URL: https://proceedings.mlr.press/v119/karimireddy20a.html.

[79] Sashank J. Reddi et al. "Adaptive Federated Optimization". In: *International Conference on Learning Representations*. 2021.

[80] Joel Stremmel and Arjun Singh. “Pretraining Federated Text Models for Next Word Prediction”. In: *Advances in Information and Communication*. Ed. by Kohei Arai. Springer International Publishing, 2021, pp. 477–488. ISBN: 978-3-030-73103-8.

[81] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[82] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing EMNLP*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/d14-1179. URL: https://aclanthology.org/D14-1179.

[83] Eugene Bagdasaryan et al. “How To Backdoor Federated Learning”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pp. 2938–2948. URL: https://proceedings.mlr.press/v108/bagdasaryan20a.html.

[84] Márk Jelasity et al. “Gossip-Based Peer Sampling”. In: *ACM Trans. Comput. Syst.* 25.3 (Aug. 2007), 8–es. ISSN: 0734-2071. DOI: 10.1145/1275517.1275520. URL: https://doi.org/10.1145/1275517.1275520.