




Enhancing Privacy and Utility in Federated Learning: A Hybrid P2P and Server-Based Approach with Differential Privacy Protection

Luca Corbucci¹^a, Anna Monreale¹^b and Roberto Pellungrini²^c

¹University of Pisa, Italy

²Scuola Normale Superiore, Italy

Keywords: Privacy Preserving Machine Learning, Federated Learning.

Abstract: Federated Learning has been recently adopted in several contexts as a solution to train a Machine Learning model while preserving users' privacy. Even though it avoids data sharing among the users involved in the training, it is common to use it in conjunction with a privacy-preserving technique like DP due to potential privacy issues. Unfortunately, often the application of privacy protection strategies leads to a degradation of the model's performance. Therefore, in this paper, we propose a framework that allows the training of a collective model through Federated Learning using a hybrid architecture that enables clients to mix within the same learning process collaborations with (semi-)trusted entities and collaboration with untrusted participants. To reach this goal we design and develop a process that exploits both the classic Client-Server and the Peer-to-Peer training mechanism. To evaluate how our methodology could impact the model utility we present an experimental analysis using three popular datasets. Experimental results demonstrate the effectiveness of our approach in reducing, in some cases, up to 32% the model accuracy degradation caused by the use of DP.


1 INTRODUCTION


Machine Learning (ML) methodologies require data to train models. Getting data is not always easy: users may not be prone to share their data with a third party. Moreover, regulations like the GDPR¹ in Europe, could set some limits on the use and sharing of the data. A key idea in such regard is to train an ML model cooperatively maintaining full control of their data. The most famous cooperative learning technique is Federated Learning (FL) (McMahan et al., 2017), which aims at training a global model by exploiting the cooperation of different participants who do not have to share their data with the other parties. In any FL architecture, each participant locally trains a model that is then shared with the other participants to achieve a global model consensus. The aggregation of the models could happen on a central server (server-based) that acts as an orchestrator or on the other clients of the network in the case of a peer-to-


peer (P2P) architecture.

Despite FL being designed to protect users' privacy, several studies have shown that the model trained with an FL architecture with a centralised aggregation may have some privacy issues (Bouacida and Mohapatra, 2021), (Melis et al., 2018), (Geiping et al., 2020). Indeed, the central aggregator has to be treated as a *trusted entity*, i.e., it has to aggregate all model updates in a correct way without discriminating against any of the clients. Moreover, it does not have to conduct any malicious attack against the participants such as membership inference attacks (Shokri et al., 2016) using the received participants' model weights. The gold standard to reduce the privacy risk of the clients involved in the FL process is the use of Differential Privacy (DP) (Dwork, 2006) during the model training.

In this paper, we propose and analyze a hybrid and private FL framework by combining both P2P and server-based architectures. In particular, we consider a scenario where the clients, before starting the model training with a server-based FL, can collaborate with some other (semi-)trusted clients to agree on a shared model. We assume that each client gains access to both a *low-privacy-risk* dataset, for which the client has no particular privacy constraints and, a

^a <https://orcid.org/0000-0001-5427-5518>

^b <https://orcid.org/0000-0001-8541-0284>

^c <https://orcid.org/0000-0003-3268-9271>

¹EU General Data Protection Regulation (GDPR) can be found at the following link: <http://bit.ly/1TlgbjI>.

high-privacy-risk dataset, which may have been recently collected, on which the client is obliged to guarantee stricter privacy requirements. Given the potential evolution of the data over time, we also consider the possibility that the second dataset might have a different distribution with respect to the first one. The approach proposed in this paper exploits the presence of these two datasets to reduce the impact of DP on the utility of the model (measured by the model's accuracy). In particular, the scenario that we analyze is a mix of P2P and server-based FL where the clients are first grouped into different clusters. In each cluster, the clients use their *low-privacy-risk* datasets to train a federated model with or without using DP. Subsequently, the clients exploit the *high-privacy-risk* datasets at their disposal for starting a differentially private server-based FL computation. Each client exploits the model already trained within the cluster to start the server-based FL training with the goal of reducing the impact of DP on the model's utility. We experimented our hybrid methodology on three well-known datasets. Experimental results show the effectiveness of our approach in reducing considerably the accuracy of the learned model.

The paper is structured as follows. In Section 2 we briefly present the related works and formalize the definition of Federated Learning and DP that we used in our methodology. Section 3 describes the details of our methodology and Section 4 presents the experiments used to validate it. Finally, Section 5 concludes the paper and presents our ideas for future developments of the proposed approach.

We summarize here our main contributions:

- We propose a new hybrid framework for training FL models that allows clients to combine the server-based and the P2P architecture;
- The framework allows the clients to adopt DP when it is needed limiting as much as possible its use to degrade as little as possible the accuracy.
- We tested our framework assuming no privacy need in the P2P computation and a differentially private server-based FL.
- We tested our approach using three different tabular datasets with different non-iid distributions, three different requirements for privacy and three different dataset configurations with respect to *low-privacy-risk* and *high-privacy-risk* dataset sizes.

2 BACKGROUND AND RELATED WORKS

Federated Learning: FL is a technique introduced in (McMahan et al., 2016) to train ML models collaboratively while preserving users' privacy. There exist two main FL architectures: server-based (McMahan et al., 2016) and peer-to-peer (P2P) (Tang et al., 2018) (Vanhaesebrouck et al., 2016) (Kasturi et al., 2022) (Sun et al., 2021). In the server-based architecture, we have two main actors during federated training: the K clients that own the training data and a central server that orchestrates the training process. The process is initiated by the central server. Each client trains a model using its own data and transmits the model updates to a central server. The server combines all model updates received and sends back the combined model updates to the clients. The process iterates until the model reaches a point of convergence in terms of model performance or until the maximum number of iterations is achieved. Instead, in a P2P architecture, communication with the server is replaced with peer-to-peer communication between the clients. All the clients in the network start training a model using their own data. At the end of the local training step, all the clients broadcast their local model update to the other clients in the network. All the clients receive the updates and aggregate them into a new model. Depending on the topology of the P2P network, this process may differ.

In addition to server-based and P2P architectures, other collaborative models have been proposed in the past few years. (Augenstein et al., 2022) presents the idea of Mixed FL. They want to exploit both the "decentralised" data owned by the clients and the "centralised" data stored on the server or in another data centre. Based on the number of clients involved in the training, we can have a cross-silo or a cross-device Federated Learning. In the cross-silo approach, only a few clients are involved in the training and all of them are always available during the training. Instead, in the cross-device approach, millions of clients are involved in the training and they are available for the training only if some conditions are met.

In terms of aggregation, there are two main algorithms that can be used by the server: Federated-SGD (FedSGD) and Federated Average (FedAvg) (McMahan et al., 2016). FedSGD is the baseline algorithm. In this approach, the server initializes a model θ_0 and shares it with the χ clients selected for the FL round. Each of the $k \in \chi$ clients performs a single gradient update step $g_k = \nabla \mathcal{L}(\theta_k, b_i)$ on the batch b_i and then shares the gradient g_k with the server. Then the server aggregates the χ gradients into a single one, updates

the model and starts a new FL round. The constraint on the single gradient update, which each client can perform before sharing it with the server, is a huge limit on the training performance. FedAvg is a more advanced algorithm that aims to improve the performance of FedSGD. In this case, the clients can perform more than a single local epoch before communicating with the server. Instead of sending the gradient to the server, with FedAvg, each client performs multiple steps of gradient descent on its own datasets and then shares the model Θ_k with the server. The server performs the aggregation computing the mean of the received models:

$$\theta_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k \quad (1)$$

In this paper, we assume the use of FedAvg as an aggregation algorithm.

Differential Privacy: Differential Privacy is a privacy-preserving model introduced by Cynthia Dwork (Dwork et al., 2006b) (Dwork and Roth, 2014). More formally, given $\epsilon > 0$ and $\delta \in [0, 1]$ a randomized mechanism \mathcal{M} satisfies (ϵ, δ) -DP if for any two “neighbouring” datasets D, D' which differ in exactly one entry and for each $S \subset \mathcal{O}$ we have $P[\mathcal{M}(D) \in S] \leq e^\epsilon P[\mathcal{M}(D') \in S] + \delta$. Intuitively, if we run the same algorithm \mathcal{M} on the two datasets, DP ensures that the two outputs will be indistinguishable up to an upper limit ϵ known as privacy budget. For the accounting of the privacy budget we use the accountant available in Opacus (Yousefpour et al., 2021), the most popular library to train Differentially Private models using Pytorch (Paszke et al., 2019). This accountant uses Rényi Differential Privacy (RDP) (Mironov, 2017), a relaxation of the original DP definition based on the Rényi divergence.

Privacy Preserving ML: The definition of DP we provided in the previous section can also be applied during the training of a ML model. A modified version of the Stochastic Gradient Descent algorithm called DP-SGD has been proposed in the literature (Abadi et al., 2016) to apply DP during the training of an ML model. The DP-SGD algorithm performs two steps. First of all, the gradient computed during the model training is clipped so that all the samples will have the same influence on the model update. Then, some noise sampled from a Gaussian distribution is added to the gradient to introduce privacy. The pseudocode of the algorithm is reported in Algorithm 1.

When applying DP to ML models, there exist two possible alternatives: Example-level DP or User-level DP. With example-level DP, we ensure that adding or removing a single training example will only change the output distribution in a minimal way. In the user-level DP instead, the output distribution will not

change even if we add or remove all the samples from one of the users. In this paper, we consider the Example-level DP. The downside of the use of DP is that it could harm the utility of the model. Therefore, it is important to consider a trade-off between model utility and privacy offered by privacy-preserving ML methods. The trade-off between privacy and utility in ML has been studied in several settings, exploring various solutions both from the algorithmic standpoint and the evaluation aspect (Zhao et al., 2020).

Algorithm 1: Differentially private SGD.

Input: Examples $\{x_1, \dots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t , noise scale σ , group size L , gradient norm bound C .

Initialize θ_0 randomly.

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $g_t(x_i) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i)$

Clip gradient

$\tilde{g}_t(x_i) \leftarrow g_t(x_i) / \max(1, \frac{\|g_t(x_i)\|_2}{C})$

Add noise

$\tilde{g}_t \leftarrow \frac{1}{L} (\sum_i \tilde{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I))$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{g}_t$

end for

return θ_T and compute the overall privacy cost (ϵ, σ) using a privacy accounting method.

Privacy Mitigation Techniques in FL: Despite having been introduced to solve privacy problems, FL has some vulnerabilities that could be exploited by an attacker. Bouacida and Mohapatra (Bouacida and Mohapatra, 2021) present a comprehensive review of all the vulnerabilities in FL. The attacks that pose a threat to the FL framework are Membership Inference attack (Shokri et al., 2016), Features leakage (Melis et al., 2018) Backdoor Attacks, Inverting Gradient attacks and Adversarial samples attacks. Some countermeasures have been proposed to tackle some of these vulnerabilities and preserve user privacy. In particular, DP (Dwork, 2006) and Homomorphic Encryption have been proposed as a possible privacy-preserving technique for FL (Pejic et al., 2022). To the best of our knowledge, real-life applications of Homomorphic Encryption for Federated Learning are still infeasible because of the computational power needed to execute this protocol.

The most used technique is DP. There are two main approaches to using DP during the training of a Federated Learning model. The first one is Central

Differential Privacy (CDP) (Geyer et al., 2017). In CDP the server is in charge of adding the necessary noise to the aggregated model. This approach entails the complete trust of the clients towards the aggregator server. The second methodology is Local Differential Privacy (LDP). In this case, the local clients involved in the federated training are in charge of applying DP. There are a few approaches that can be used to apply DP to local clients. (Wei et al., 2019) proposed to apply LDP on the client side after the training of the local model. This approach could destroy the model's utility. Therefore, the most used approach is based on DP-SGD (Differentially-Private Stochastic Gradient Descent) (Abadi et al., 2016). With this method, DP is applied directly to the gradients computed during the training process. In addition to the approaches that only employ DP during the training, both (Bonawitz et al., 2017) and (Truex et al., 2019) proposed a solution to train models using FL combining the use of DP with the use of secure multi-party computation during the aggregation step.

FL Techniques Against Attacks: in (Chandrasekaran et al., 2022) Chandrasekaran et al. proposed a hierarchical FL framework that exploits DP to defend against privacy attacks like membership inference attacks and model inversion. They divide the nodes into zones and for each zone, there is a super node. The super-node is chosen by the nodes of the zone using some algorithms used in P2P networks. The idea is that the nodes in a zone communicate with a super node. Each node sends the weights and the super node averages all of them adding noise. Later, the super-nodes send the weights with noise to the central server. The nodes can add noise when they train their local model or they can trust the super-node.

FL with Clustering: Several modifications of the classic FL protocol have been proposed in the past few years. However, in the literature, the idea of clustering is usually used with FL to deal with clients with different distributions with the goal of increasing the utility of the models making them tailored to the different distributions. In particular, (Briggs et al., 2020) proposed the idea of exploiting the similarity of the local gradients of the clients to cluster them. This allows the training of independent and specialised models. This protocol was extended in (Luo et al., 2023) to adapt to the privacy-preserving context. In particular, they designed a series of secure cryptographic protocols to ensure parties' privacy. However, our modification of the standard FL protocol is different from the ones proposed in the past. First of all, we cluster the clients based on the trust among the clients. Moreover, we add a pre-training Peer-to-peer step inside the clusters before starting the classic FL proto-

col with the server. (Shenaj et al., 2023) clustered the clients based on the styles of the images belonging to each client to have models able to exploit both global and cluster-specific parameters. The idea of mixing both P2P and server-based training has been presented in (Yeganeh et al., 2022). In particular, their idea was to perform the training with the server first and then, after a few rounds, cluster the clients based on their model and then continue the training inside the clustering. Even though this idea seems similar to ours, the goals are different. In this case, they exploit the training inside the cluster to train models tailored to each cluster. Moreover, they do not consider the use of privacy mitigation techniques. This last part is crucial in our proposal because of the goal of reducing the impact of privacy mitigation on accuracy by exploiting the clusters.

The Effectiveness of Pre-Training on FL: When FL models are trained and evaluate it is common to start from a random initialization of the model. However, several studies (Chen et al., 2023) (Nguyen et al., 2023) empirically studied the impact of starting the FL process from a pre-trained model. In general, this approach has a positive effect both on the final accuracy of the trained model and on the convergence time. Moreover, it can also be helpful in reducing the effect of data heterogeneity, this is particularly useful when we consider a non-iid scenario that is typical when dealing with FL. Recently, the idea of using a public dataset to pre-train a model has also been proposed as a possible solution to reduce the degradation of the model utility due to the use of DP (Kurakin et al., 2022) (Nasr et al., 2023) (De et al., 2022). In our paper, we leverage these results to guarantee both a modularity of the training architecture and to achieve a smaller degradation of the model utility while training an FL model.

3 METHODOLOGY

The main goal of this paper is the introduction of a more customizable, modular and easy to update architecture for FL which empowers users to choose how and when to use DP in a way that does not unduly degrade the utility of the model. When DP-SGD is applied during the training of a model, privacy comes at the cost of reducing the utility of the model, our methodology aims to balance these two requirements by exploiting two different types of architecture: P2P and client-server. As in any classic FL system, our methodology considers two main actors:

- A set of K clients that take part in the FL training. These clients can cooperate with each other and to

do so they can use a P2P or server-based architecture. If they decide to use a P2P architecture, they can gather in different clusters. The belonging of a client to a cluster may depend, for instance, on a similar distribution of cluster clients on the level of trust that clients may have with each other or on the membership of the same organisation. In addition to the P2P architecture, the clients can decide to use a classic server-based FL. We assume that each client k gains access to two datasets: a *low-privacy-risk* dataset lr_k , for which the client has no special requirements in terms of privacy, and a *high-privacy-risk* data hr_k . This *high-privacy-risk* dataset is used to simulate new data collections potentially with a different distribution than the one of lr_k and with higher privacy requirements.

- A central server that orchestrates the whole learning process for learning a collective ML model by aggregating clients' models.

The scenario that we consider fits in a context in which multiple entities, belonging to different organizations have to train a model. For instance, if we consider a scenario with multiple hospitals, we could have that the ones belonging to the same region can trust each other and thus, they can collaborate in a cluster with no privacy mitigation. Then, after a cluster-based cooperation, they could start to collaborate with the other hospitals by exploiting the server coordination. The opportunity of merging both cluster-based collaboration and server-based collaboration could also enable the clients to set up different local strategies for learning. As a consequence, with our flexible architecture the clients inside a cluster could learn their local models, without any privacy mitigation strategy, using the *low-privacy-risk* dataset. Then, once learning an initial common model on these data they can use the complete and potentially up-to-date and risky dataset for the final learning, involving the server and the other hospitals. Clearly, in this last part, they have to apply the appropriate privacy mitigation strategies.

More formally, the different phases of the protocol that we propose are the following:

1. **Cluster Formation Phase:** we consider a scenario in which the clients are first gathered into clusters, and then they start the training with the P2P FL architecture. The whole set of K clients that want to participate in the training of a federated model is partitioned in C clusters. Each client $k \in K$ is assigned exclusively to a cluster C_i . Therefore, given the set of clusters $C = C_1, C_2, \dots, C_m$ we have that $\sum_{i=1}^m |C_i| = |K|$.
2. **P2P Training Phase:** During this phase the

clients of each cluster train a model using a P2P protocol. We report the pseudocode of the P2P Training Phase in Algorithm 2. At the beginning of this phase, each client initializes a model θ_k from whom to start the training (Line 1 of Algorithm 2). Then, the client of each cluster performs a number of local training epochs E_c using its own dataset lr_k (Line 3 of the algorithm) and broadcasts the model to all the other clients inside the cluster (Line 5). Next, each client waits to receive the models broadcasted by the other clients in the cluster (Lines 6 and 7). Once the client has received all the models, it can apply FedAvg to create the final model θ . The procedure is then started again from Line 2 of Algorithm 2 and repeated for a certain number of FL rounds. During this phase, the clients can decide if they want to use DP-SGD for training a *differentially private* model. In our experiments, we do not use DP during the P2P Phase because we assume trust among the clients of each cluster.

3. **Server Phase:** During this phase, the standard FL protocol is executed. The different clients in this case use the *high-privacy-risk* dataset hr_k and execute a fixed number of local training epochs E_s before sharing their local model with a server. For each FL Round t , the server S aggregates clients' models into a single collective model θ_t . If the clients do not trust the central server they can mitigate the privacy risks of the model by applying a protection mechanism. In our experiments, we rely on DP and we apply the DP-SGD algorithm.

Algorithm 2: P2P-phase (Client Side.)

Require: E_c local epochs for P2P Phase. lr_k *low-privacy-risk* data of client k . L list of the clients located in the same cluster as the client k .

- 1: Initialize model θ_k
 - 2: **for** $e \in E_c$ **do**
 - 3: Train θ_c using local data lr_k
 - 4: **end for**
 - 5: Broadcast(θ_k , len(lr_k), V)
 - 6: **for** ($dl \in L$)
 - 7: $\Theta_l, n_l \leftarrow$ Receive_model(l)
 - 8: **end for**
 - 9: $\theta \leftarrow \sum_{l=1}^{|L|} \frac{n_l}{n} \theta_l$ # Aggregate with FedAvg
-

Figure 1 shows a representation of the protocol used for our experiments. In this case, we are considering two clusters with 3 clients each. The training is first computed inside the clusters and then, it is completed using a server-based approach.

As mentioned above, the two phases of this proto-

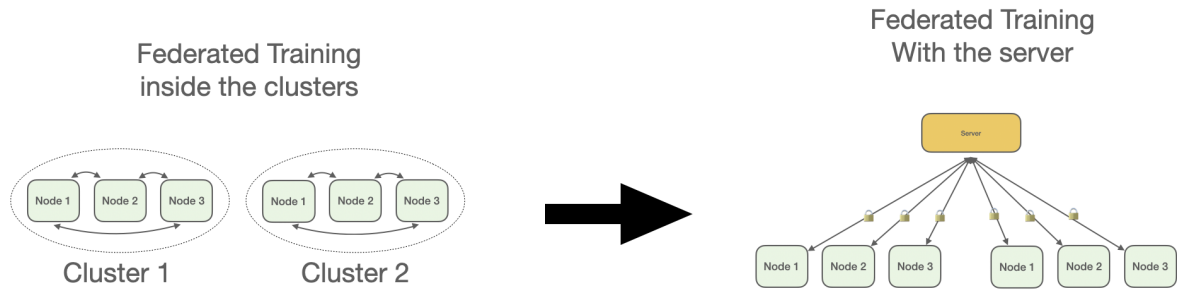


Figure 1: A representation of our protocol. The model is firstly trained inside the different clusters using a P2P architecture. Then server-based training is employed to complete the training. During this last phase, clients use DP-SGD to guarantee privacy protection during the training.

col can be composed based on the client’s preferences in order to ensure maximum flexibility and create various learning processes to train the federated model. Obviously, all the clients taking part in the training must agree on the protocol.

Privacy Protection in Our Methodology: The proposed methodology is agnostic w.r.t. the privacy mitigation mechanism. We chose DP as it gives us the possibility to select different privacy budgets at different phases of the protocol. In this paper, we rely on the DP-SGD (Abadi et al., 2016) and on the RDP Accountant (Mironov, 2017) implemented in Opacus (Yousefpour et al., 2021). The noise used to guarantee privacy is drawn from a Gaussian Distribution (Dwork et al., 2006a). Because of the composability of our framework, it is possible to apply DP-SGD both during the P2P Phase and during the server Phase. In both cases, we need to set a corresponding privacy budget that we want to guarantee. It is possible to set a privacy budget (ϵ_1, δ_1) for the P2P Phase and then a different privacy budget (ϵ_2, δ_2) for the server phase. Then, we apply the basic composition theorem (Dwork and Roth, 2014) to compute the final privacy budget guaranteed by the protocol. To do so, we sum the different privacy budgets of the two different phases into a single final privacy budget: $(\epsilon = \epsilon_1 + \epsilon_2, \delta = \delta_1 + \delta_2)$. In our experiments we do not apply DP-SGD during the P2P Training Phase but only during the server phase. Therefore, we guarantee a privacy budget (ϵ_2, δ_2) .

3.1 How We Partition the Data

A large number of previous works made a strong assumption about the i.i.d.-ness distribution of the dataset used to train a federated learning model. This is not true in a real-life scenario since each participant could have a different data distribution. Moreover, the distributions could also evolve because of new data collections. In our experiments, the goal was to simulate this scenario. Since we did not have a dataset that

was naturally split into different parts in the way we needed, we had to simulate this scenario by splitting a dataset such that it could respect these constraints. Given a dataset D , we partitioned it using the following approach:

- Given a dataset D we first split it into the C parts corresponding to the C clusters. This partitioning of the samples into clusters is based on the sample target. To generate imbalanced distributions, we used a Dirichlet distribution for each of the targets. Then, based on these distributions, we assigned the corresponding quantity of data to each of the clients. The Dirichlet distribution has a parameter α that determines the level of non-i.i.d.-ness. In our experiments, we used $\alpha = 0.5$ to simulate a highly skewed dataset.
- The procedure used to partition the data into C clusters is then recursively applied inside each of the C clusters. In particular, given a cluster C_i with K_i clients, the dataset D_i assigned to cluster C_i is partitioned into K_i parts using the same Dirichlet distribution approach with the same α parameter.
- The last step is the creation of the *high-privacy-risk* and *low-privacy-risk* datasets for each of the clients of the different clusters. For each client k of cluster C_i , the dataset D_k is partitioned into 2 parts using the Dirichlet Distribution with $\alpha = 0.5$. This is done to simulate the presence of a *low-privacy-risk* dataset and a *high-privacy-risk* dataset with different data distributions. As we already said, in our experiments, the *low-privacy-risk* dataset will be used for the intra-cluster computations while the *high-privacy-risk* dataset will be used for completing the training of the final model. The idea is to simulate a situation in which clients in a cluster can exploit an old and/or small dataset with low privacy requirements for an initial training and then, can complete the collaborative training by exploiting the up-to-date and complete dataset.

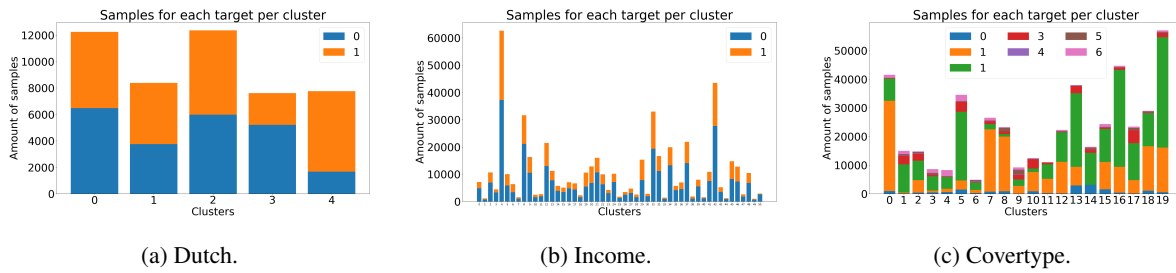


Figure 2: Fig. 2a, 2b and 2c show the result of the dataset partition among the clusters. In the case of Covertypes (Fig 2c) that is a multiclass dataset with 7 possible targets, we observe a more “real-life” distribution in which some client only receives data with some of the possible targets.

4 EXPERIMENTS

We evaluate our methodology using three different tabular datasets: Dutch (Van der Laan, 2001), Covertypes (Blackard, 1998) and Income (Ding et al., 2022). Dutch is a dataset collected in the Netherlands that contains 60.420 samples. The task is to predict whether the individuals have a salary higher or lower than 50K. Covertypes is a multi-class dataset that contains 581.012 samples. The task is to predict one of the seven possible forest cover types. Income is a bigger version of the popular Adult dataset (Becker and Kohavi, 1996). It is composed of 1.664.500 samples and it contains census data about 50 states of the United States of America and Puerto Rico. As with Dutch, even in this case, the task is to predict if the individuals earn more or less than 50K.

To guarantee a comprehensive evaluation of the proposed methodology, we considered multiple settings in terms of privacy requirements and size of the *low-privacy-risk* and *high-privacy-risk* datasets. In terms of privacy requirements, for each dataset, we considered three different privacy budgets:

- With Dutch we used $(\epsilon = 0.2, \delta = 9 \times 10^{-3})$, $(\epsilon = 0.5, \delta = 9 \times 10^{-3})$ and $(\epsilon = 1, \delta = 9 \times 10^{-3})$
- With Income we used $(\epsilon = 0.5, \delta = 1 \times 10^{-2})$, $(\epsilon = 1.0, \delta = 1 \times 10^{-2})$ and $(\epsilon = 2.0, \delta = 1 \times 10^{-2})$
- With Covertypes we used $(\epsilon = 0.5, \delta = 3 \times 10^{-3})$, $(\epsilon = 1.0, \delta = 3 \times 10^{-3})$ and $(\epsilon = 2.0, \delta = 3 \times 10^{-3})$

The δ parameter is computed as $\delta = \max_{k=1}^K \frac{1}{n_k}$ where n_k is the size of the dataset of client k . The selection of both ϵ and δ was guided by recommendations found in the literature (Ponomareva et al., 2023).

For each dataset, we set a specific configuration in terms of the number of clients and clusters exploiting the different sizes of the datasets:

- with Dutch we created 5 clusters with 15 clients each. In total, we have 75 clients;

- with Income we exploited the natural division of the dataset into 51 states creating 51 clusters. For each of these clusters, we used 5 clients. In total, we have 255 clients.
- with Covertypes we considered 20 clusters with 5 clients each. In total, we have 100 clients

After applying the partition strategy among clusters, described in Section 3.1, we obtain a data distribution for each dataset as depicted in Figure 2. We observe that for Income (Figure 2b) and Covertypes (Figure 2c) data distributions are more interesting with respect to their non-i.i.d.-ness. Differently from Dutch and Income, Covertypes is a multiclass dataset with 7 possible targets. This feature of the dataset allows us to have a more “real-life” distribution of the data in which some cluster only sees a portion of the possible targets. Income distribution, instead, is interesting because it is “naturally” split into 51 clusters corresponding to the USA states and Puerto Rico.

Lastly, in our experiments, we also compared different sizes of the *low-privacy-risk* and *high-privacy-risk* datasets:

- 20% *low-privacy-risk* and 80% *high-privacy-risk*;
- 30% *low-privacy-risk* and 70% *high-privacy-risk*;
- 40% *low-privacy-risk* and 60% *high-privacy-risk*.

To choose the hyperparameters for the experiments we performed a hyperparameter tuning. To do this, we split the *low-privacy-risk* and the *high-privacy-risk* dataset of each client into two parts, one used to train the model (80%) and one used to validate it (20%). The model used for the experiments is a Logistic Regressor implemented with Pytorch using a simple neural network with a single linear layer.

4.1 Results

In this section, we present the result of the application of our methodology. In particular, we want to highlight the impact of using a *low-privacy-risk* dataset

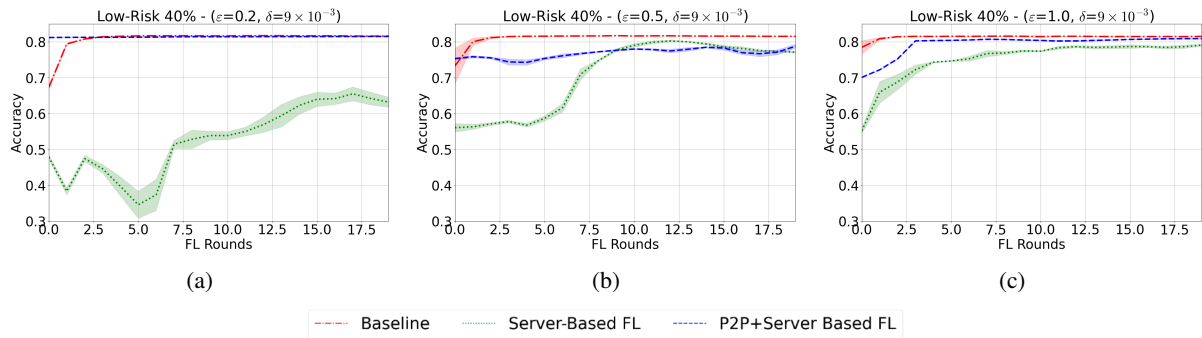


Figure 3: Figures 3a, 3b and 3c show a comparison between a Baseline (red line), the model trained with server-based architecture on the *high-privacy-risk* dataset using DP-SGD (green line) and the model pre-trained in the clusters and then trained with the server-based architecture (blue line) when using the **Dutch Dataset**.

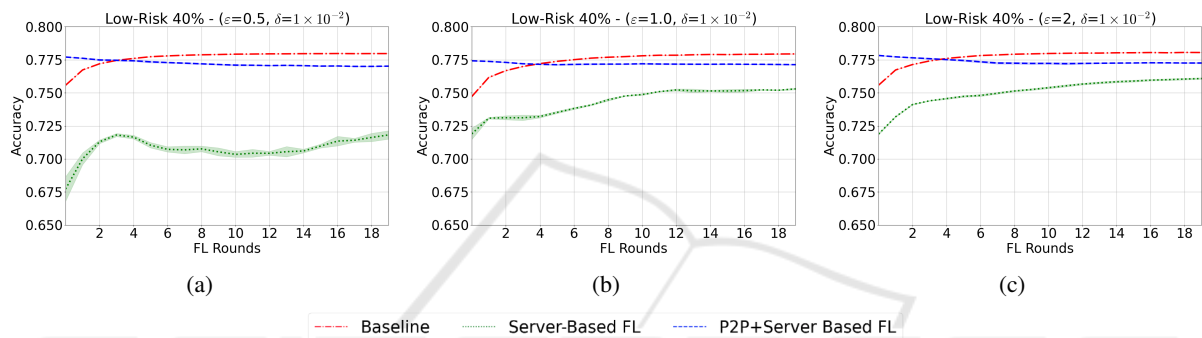


Figure 4: Figures 4a, 4b and 4c show a comparison between a Baseline (red line), the model trained with server-based architecture on the *high-privacy-risk* dataset using DP-SGD (green line) and the model pre-trained in the clusters and then trained with the server-based architecture (blue line) when using the **Income Dataset**.

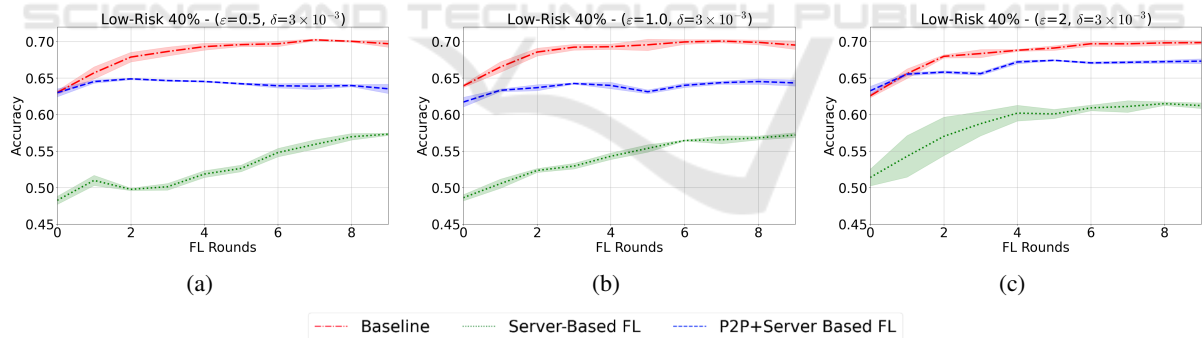


Figure 5: Figures 5a, 5b and 5c show a comparison between a Baseline (red line), the model trained with server-based architecture on the *high-privacy-risk* dataset using DP-SGD (green line) and the model pre-trained in the clusters and then trained with the server-based architecture (blue line) when using the **Coverttype Dataset**.

during the P2P Training Phase on improving the final model accuracy while guaranteeing privacy protection. For each experiment, we report the average performance and the standard deviation of 4 different runs, each with a different seed. In our experiments, we compare the following models:

- “Baseline”: A model trained using a classic server-based FL where clients use the *high-privacy-risk* dataset. In this case, clients do not

apply any DP protection (Red Line in the plots).

- “Server-Based FL”: A differentially private model trained with server-based FL where clients use the *high-privacy-risk* dataset (Green Line in the plots)
- “P2P + Server-Based FL”: The outcome of our methodology. In this case, we pre-train models inside multiple clusters using the clients’ *low-privacy-risk* datasets without DP. Then, we train a global model with a classic server-based FL,

where pre-trained models are refined using the client's *high-privacy-risk* datasets using DP (Blue Line in the plots).

Table 1: Dutch Dataset.

Low-Privacy-Risk	(ϵ, δ)	Accuracy FL	Accuracy P2P+FL	Improvement
20%	-	0.817 + 0.001	-	-
30%	-	0.815 + 0.005	-	-
40%	-	0.810 + 0.003	-	-
20%	$(0.2, 9 \times 10^{-3})$	0.629 + 0.06	0.81 + 0.003	22.34%
20%	$(0.5, 9 \times 10^{-3})$	0.792 + 0.001	0.747 + 0.003	-%
20%	$(1, 9 \times 10^{-3})$	0.785 + 0.003	0.808 + 0.001	-%
30%	$(0.2, 9 \times 10^{-3})$	0.611 + 0.018	0.808 + 0.001	32.24%
30%	$(0.5, 9 \times 10^{-3})$	0.705 + 0.017	0.682 + 0.005	-%
30%	$(1, 9 \times 10^{-3})$	0.788 + 0.002	0.803 + 0.001	1.86%
40%	$(0.2, 9 \times 10^{-3})$	0.632 + 0.014	0.815 + 0.001	22.45%
40%	$(0.5, 9 \times 10^{-3})$	0.771 + 0.002	0.788 + 0.006	2.15%
40%	$(1, 9 \times 10^{-3})$	0.791 + 0.002	0.809 + 0.001	2.22%

Table 2: Income Dataset.

Low-Privacy-Risk	(ϵ, δ)	Accuracy FL	Accuracy P2P+FL	Improvement
20%	-	0.781 + 0.000	-	-%
30%	-	0.780 + 0.000	-	-%
40%	-	0.779 + 0.000	-	-%
20%	$(0.5, 1 \times 10^{-2})$	0.732 + 0.003	0.769 + 0.000	4.81%
20%	$(1, 1 \times 10^{-2})$	0.745 + 0.001	0.777 + 0.000	4.11%
20%	$(2, 1 \times 10^{-2})$	0.756 + 0.000	0.777 + 0.000	2.77%
30%	$(0.5, 1 \times 10^{-2})$	0.741 + 0.009	0.757 + 0.001	2.15%
30%	$(1, 1 \times 10^{-2})$	0.757 + 0.000	0.774 + 0.001	2.19%
30%	$(2, 1 \times 10^{-2})$	0.765 + 0.001	0.757 + 0.000	-%
40%	$(0.5, 1 \times 10^{-2})$	0.718 + 0.003	0.770 + 0.000	6.75%
40%	$(1, 1 \times 10^{-2})$	0.753 + 0.001	0.771 + 0.000	2.20%
40%	$(2, 1 \times 10^{-2})$	0.761 + 0.001	0.773 + 0.001	1.55%

Table 3: Coverttype Dataset.

Low-Privacy-Risk	(ϵ, δ)	Accuracy FL	Accuracy P2P+FL	Improvement
20%	-	0.698 + 0.002	-	-
30%	-	0.697 + 0.005	-	-
40%	-	0.695 + 0.006	-	-
20%	$(0.5, 3 \times 10^{-3})$	0.554 + 0.003	0.594 + 0.005	7.22%
20%	$(1, 3 \times 10^{-3})$	0.614 + 0.003	0.642 + 0.002	4.56%
20%	$(2, 3 \times 10^{-3})$	0.64 + 0.004	0.652 + 0.002	1.87%
30%	$(0.5, 3 \times 10^{-3})$	0.533 + 0.004	0.632 + 0.002	18.57%
30%	$(1, 3 \times 10^{-3})$	0.615 + 0.003	0.627 + 0.006	1.95%
30%	$(2, 3 \times 10^{-3})$	0.612 + 0.004	0.616 + 0.004	0.65%
40%	$(0.5, 3 \times 10^{-3})$	0.573 + 0.002	0.635 + 0.006	10.82%
40%	$(1, 3 \times 10^{-3})$	0.572 + 0.003	0.643 + 0.004	12.41%
40%	$(2, 3 \times 10^{-3})$	0.612 + 0.001	0.673 + 0.003	9.96%

In Tables 1, 2 & 3 we report the results of the experiments conducted on the three datasets with the different combinations of DP parameters and *low-privacy-risk* vs *high-privacy-risk* datasets. In the tables, we compare the accuracy achieved with “Server-Based FL” with the one achieved with our methodology “P2P+Server Based FL” highlighting the improvement in percentage. We observe that, regardless of the *low-privacy-risk* dataset size, we have a higher improvement in performance in all the datasets when using DP with the lowest privacy parameters. This improvement is more evident for Dutch and Coverttype. This means that our approach enables high levels of privacy protection while guaranteeing very good performance of the final FL model. Moreover, it is also important to note that the accuracy of the models is very close to the one obtained by applying the standard FL computation without any privacy requirements (first three lines of each table). Clearly, the size of the *low-privacy-risk* and *high-privacy-risk*

datasets also impacts the improvement that we can achieve with our methodology. In general, we observe that the use of a *low-privacy-risk* dataset with 20% or 30% of samples combined with looser privacy requirements reduces our gain in performance with respect to the classic Server-based FL. In some cases, as for Dutch, it also makes it less effective. However, when we increase the size of the *low-privacy-risk* dataset, our methodology can bring an improvement to the final model accuracy that, in the case of Dutch, ranges from 22%, when using a privacy requirement $(\epsilon = 0.2, \delta = 9 \times 10^{-3})$, to 2%, when using $(\epsilon = 1, \delta = 9 \times 10^{-3})$.

Concerning these experiments, we also report in Figures 3, 4 and 5 the evolution of model accuracy over the different FL rounds. All the plots refer to the results that we obtained using data partitioning into 40% *low-privacy-risk* and 60% *high-privacy-risk* on each client. To show the impact of the privacy requirement, we report all three different privacy requirements per dataset. We observe that the pre-training inside the clusters of our methodology helps both in starting the “P2P+Server Based FL” training from a more accurate model and, in reducing the final accuracy degradation with respect to the Server-Based FL. Observing the different rounds we can notice that our approach allows us to converge to the final model accuracy faster and the model accuracy becomes stable over the FL rounds faster regardless of the privacy budget. Looking at the three datasets we observe that in Dutch and Income (Figure 3 and 4) the decrease in privacy protection leads to a decrease in the improvement of our methodology. For example, in Dutch, where we used a severe privacy budget this is more evident. Indeed the improvement goes from a 22.45% of the case with $(\epsilon = 0.2, \delta = 9 \times 10^{-3})$ to the 2.22% in the case with $(\epsilon = 1, \delta = 9 \times 10^{-3})$.

In general, we can observe that the more is the impact of DP on the model trained by a server-based FL the higher the improvement offered by our methodology. This is because our methodology, thanks to the pre-training is more resistant to the potential model degradation caused by DP. In Coverttype (Figure 5c) the decrease of the improvement is less evident because the effect of DP on the “Server-Based FL” experiments with the three privacy settings is not markedly different. Indeed, the improvement ranges from 10.82%, in the case of $(\epsilon = 0.5, \delta = 3 \times 10^{-3})$, to 9.96, in the case of $(\epsilon = 2, \delta = 3 \times 10^{-3})$.

5 CONCLUSION

In this paper, we introduced a new hybrid Federated Learning architecture that mixes both aspects of P2P and Server-based FL training to reduce the impact of DP on the final model accuracy. Our idea is to compose the classic FL architecture, in which the model is trained using a server that acts as an orchestrator, with a peer-to-peer architecture, in which the model is trained without a third-party server. In our scenario, each client possesses both a *low-privacy-risk* and a *high-privacy-risk* dataset used during the P2P and the Server-based Phase, respectively. We tested our methodology using three different tabular datasets, three different privacy requirements and three different sizes of *low-privacy-risk* and *high-privacy-risk* datasets. Our experiments show how our methodology can be effective in reducing the impact of DP on the model's accuracy. When comparing the final model accuracy of the standard Server-based FL approach with our methodology, we were able to achieve in some cases an improvement up to 32%. In particular, we found that the impact of our methodology is more evident when using high privacy requirements, i.e. low privacy budget, which typically negatively impacts the standard Server-based FL. The size of the *low-privacy-risk* datasets is another important factor to consider, the more we increase it the more our methodology is effective. As a future work, we intend to test a wider range of architectures mixing the two building blocks, P2P-based and server-based training, in different ways. Moreover, we would like to validate the methodology on more datasets of different modalities.

ACKNOWLEDGMENT

This work is partially supported by the PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", funded by the European Commission under the NextGeneration EU programme, PNRR -"SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013, TANGO - Grant Agreement no. 101120763, H2020-INFRAIA-2019-1: Res. Infr. G.A. 871042 *SoBigData++*

REFERENCES

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep

learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM.

Augenstein, S., Hard, A., Ning, L., Singhal, K., Kale, S., Partridge, K., and Mathews, R. (2022). Mixed federated learning: Joint decentralized and centralized learning.

Becker, B. and Kohavi, R. (1996). Adult. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5XW20>.

Blackard, J. (1998). Covertype. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C50K5N>.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17*, page 1175–1191, New York, NY, USA. Association for Computing Machinery.

Bouacida, N. and Mohapatra, P. (2021). Vulnerabilities in federated learning. *IEEE Access*, 9:63229–63249.

Briggs, C., Fan, Z., and Andras, P. (2020). Federated learning with hierarchical clustering of local updates to improve training on non-iid data.

Chandrasekaran, V., Banerjee, S., Perino, D., and Kourtellis, N. (2022). Hierarchical federated learning with privacy.

Chen, H.-Y., Tu, C.-H., Li, Z., Shen, H.-W., and Chao, W.-L. (2023). On the importance and applicability of pre-training for federated learning.

De, S., Berrada, L., Hayes, J., Smith, S. L., and Balle, B. (2022). Unlocking high-accuracy differentially private image classification through scale.

Ding, F., Hardt, M., Miller, J., and Schmidt, L. (2022). Retiring adult: New datasets for fair machine learning.

Dwork, C. (2006). Differential Privacy. In *Automata, Languages and Programming*, pages 1–12. Springer, Berlin, Germany.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*, pages 486–503. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer.

Dwork, C. and Roth, A. (2014). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. (2020). Inverting gradients – how easy is it to break privacy in federated learning?

- Geyer, R. C., Klein, T., and Nabi, M. (2017). Differentially private federated learning: A client level perspective.
- Kasturi, A., Sivaraju, R., and Hota, C. (2022). Fedpeer: A peer-to-peer learning framework using federated learning. In Patgiri, R., Bandyopadhyay, S., Borah, M. D., and Emilia Balas, V., editors, *Edge Analytics*, pages 517–525, Singapore. Springer Singapore.
- Kurakin, A., Song, S., Chien, S., Geambasu, R., Terzis, A., and Thakurta, A. (2022). Toward training at imagenet scale with differential privacy.
- Luo, S., Fu, S., Luo, Y., Liu, L., Deng, Y., and Wang, S. (2023). Privacy-preserving federated learning with hierarchical clustering to improve training on non-iid data. In Li, S., Manulis, M., and Miyaji, A., editors, *Network and System Security*, pages 195–216, Cham. Springer Nature Switzerland.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *AIS-TATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.
- McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2016). Communication-efficient learning of deep networks from decentralized data.
- Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. (2018). Exploiting unintended feature leakage in collaborative learning.
- Mironov, I. (2017). Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE.
- Nasr, M., Mahloujifar, S., Tang, X., Mittal, P., and Houmansadr, A. (2023). Effectively using public data in privacy preserving machine learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25718–25732. PMLR.
- Nguyen, J., Wang, J., Malik, K., Sanjabi, M., and Rabbat, M. (2023). Where to begin? on the impact of pre-training and initialization in federated learning.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library.
- Pejic, I., Wang, R., and Liang, K. (2022). Effect of homomorphic encryption on the performance of training federated learning generative adversarial networks.
- Ponomareva, N., Hazimeh, H., Kurakin, A., Xu, Z., Denison, C., McMahan, H. B., Vassilvitskii, S., Chien, S., and Thakurta, A. G. (2023). How to dp-fy ml: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research*, 77:1113–1201.
- Shenaj, D., Fani, E., Toldo, M., Caldarola, D., Tavera, A., Michieli, U., Ciccone, M., Zanuttigh, P., and Caputo, B. (2023). Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 444–454.
- Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2016). Membership inference attacks against machine learning models.
- Sun, T., Li, D., and Wang, B. (2021). Decentralized federated averaging.
- Tang, H., Lian, X., Yan, M., Zhang, C., and Liu, J. (2018). D²: Decentralized training over decentralized data.
- Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning.
- Van der Laan, P. (2001). *The 2001 Census in the Netherlands: Integration of Registers and Surveys*.
- Vanhaesebrouck, P., Bellet, A., and Tommasi, M. (2016). Decentralized collaborative learning of personalized models over networks.
- Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farhad, F., Jin, S., Quek, T. Q. S., and Poor, H. V. (2019). Federated learning with differential privacy: Algorithms and performance analysis.
- Yeganeh, Y., Farshad, A., Boschmann, J., Gaus, R., Frantzen, M., and Navab, N. (2022). Fedap: Adaptive personalization in federated learning for non-iid data. In Albarqouni, S., Bakas, S., Bano, S., Cardoso, M. J., Khanal, B., Landman, B., Li, X., Qin, C., Rekik, I., Rieke, N., Roth, H., Sheet, D., and Xu, D., editors, *Distributed, Collaborative, and Federated Learning, and Affordable AI and Healthcare for Resource Diverse Global Health*, pages 17–27, Cham. Springer Nature Switzerland.
- Yousefpoor, A., Shilov, I., Sablayrolles, A., Testuggine, D., Prasad, K., Malek, M., Nguyen, J., Ghosh, S., Bharadwaj, A., Zhao, J., Cormode, G., and Mironov, I. (2021). Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*.
- Zhao, B. Z. H., Kaafar, M. A., and Kourtellis, N. (2020). Not one but many tradeoffs. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*. ACM.