Decentralized Federated Learning for Nonintrusive Load Monitoring in Smart Energy Communities

Alessandro Giuseppi[1,*], Sabato Manfredi[2], Danilo Menegatti [1], Antonio Pietrabissa[1] and Cecilia Poli[3]

***Abstract*— Federated Learning is a distributed learning solution for machine learning problems without the need of collecting the available data in a single centralized data centre. With the standard FL approaches, model training is performed locally and a centralized server collects and elaborates the trainable parameters of the local models: even if data are not shared, the presence of the centralized server still rises trust and security issues. In this work, we introduce the Decentralized Federated Learning (DECFEDAVG) algorithm, which aims at achieving complete decentralization by the lack of a coordination server, and compare its performance against the original federated learning algorithm Federated Averaging (FEDAVG) over the Nonintrusive Load Monitoring problem.**

***Index Terms*— Federated Learning, Deep Neural Networks, Distributed Systems, Nonintrusive Load Monitoring, Demand Side Management.**

I. INTRODUCTION

In Federated Learning (FL) a federation of clients learns a model by coordinating with a global server, without the need to share their own local data. Thanks to its privacy-preserving nature, it finds applications in distributed scenarios, when data cannot be shared with a centralized entity. Given a federation of a certain number of clients, the goal of FL is to let them cooperate by sharing their knowledge while avoiding any data exchange. In particular, during each *communication round*, the model of the coordinating server is updated by averaging the *trainable parameters* of the models of the federated clients, which are trained on their corresponding *local data* [1]. For the algorithm to work, it is required the centralized server to be trusted by the federation; if on the one hand this provides a communication-efficient solution with privacy guarantees, on the other hand it may be prone to malicious attacks and constitutes a single point of failure because the clients of the federation cannot communicate with one another, thus cannot verify its behaviour. A possible solution to this problem is to consider decentralized federations with point-to-point client agreements, specifically with sparse communication tolopogies, as depicted in Figure 1, more robust to malicious attacks; hence to study FL algorithms able to cope with them.

[1] Dept. of Computer, Control, and Management Engineering (DIAG), University of Rome "La Sapienza", Rome, Italy

[2] Dept. of Electrical Engineering and Information Technology, University of Naples "Federico II", Naples, Italy

[3] National Centre for Innovative Technologies in Public Health (TISP), Istituto Superiore di Sanità, Rome, Italy

This study has been partially supported by the project "Avvio alla ricerca 2021" of the University of Rome "La Sapienza" and by the Lazio region, in the scope of the project FedMedAI, POR FESR Lazio 2014 – 2020 (Azione 1.2.1), Prot. n. A0375-2020-36491 - 23/10/2020

*Corresponding author: giuseppi@diag.uniroma1.it

In this work, we propose the Decentralized FL algorithm, DECFEDAVG, a *decentralized* extension of FEDAVG [1] able to deal with group of federated entities characterized by sparse and arbitrary communication agreements. DecFedAvg was designed in the scope of the project FedMedAI and was tailored for the privacy requirements imposed by the EU General Data Protection Regulation (GDPR) [2] on personal data, with particular focus on the healthcare domain. The highlights and main contributions of this work are:

- The idea to extend FL algorithms to a fully decentralized setting that does not require a coordinating server;
- The proposal of a decentralized version of the original algorithm FEDAVG, named Decentralized Federated Averaging DECFEDAVG;
- Validating examples are discussed, demonstrating the applicability and performance properties of the proposed algorithm on the Nonintrusive Load Monitoring (NILM) problem.

The reminder of the paper is organized as follows: Section II discusses the relevant works in the literature. Section III presents the proposed algorithm. Section IV describes the dataset used for testing. Finally, Section V reports the results of our tests and Section VI draws the conclusions and discusses some future works.

II. RELATED WORKS

The concept of FL and its first ever algorithm (FEDAVG) were proposed for the first time in [1], [3] to address the problem of smartphone cooperation without disclosing any data from their users. The main novelty of FL was the capability to deal with data partitioned *as it is*, namely close to its sources, without the need to perform any redistribution and coordinated split, as in standard distributed learning solutions. For its privacy-preserving properties, FL finds application in multiple fields [4], such as 3D connectivity-based heterogeneous networks enabled by aerial drones [5], Industry 4.0 [6], healthcare [7], [8], IoT/Edge Computing setting [9], [10], power systems [11]. In particular, with respect to the latter, it finds application in the NILM problem [12], an approach to monitor individual appliances' electrical power consumption from aggregate measurements.

Starting from the original formulation, there have been multiple versions [13]–[15], with the aim of enhance privacy and security [16]–[18] to prevent direct or indirect data leakage [19], [20], and on reducing the communication cost associated to the distributed training [21], [22]. Regardless of the particular considered algorithm, the capability to deal with data distributed in a non-IID and imbalanced way over

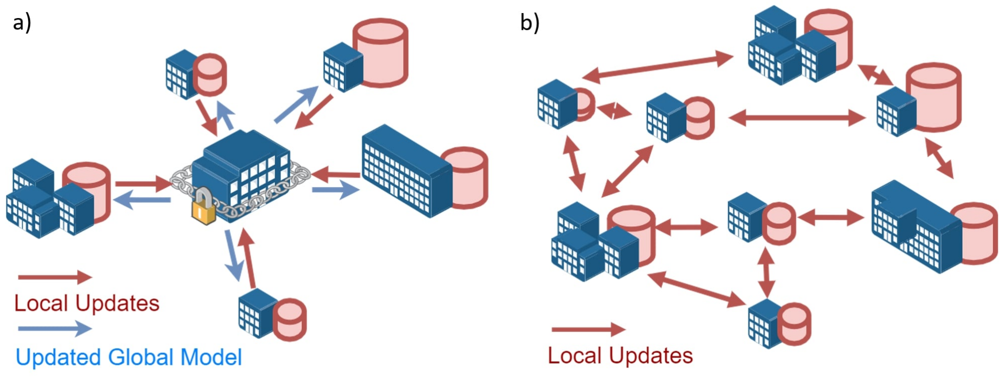

Fig. 1: (a) Standard FL architecture (left); (b) Example of a decentralized federation with bi-directional point-to-point communications (right)

the clients remains the main feature, shared with the original formulation; the distributed nature of data sources is, most of the time, a reflection of their geographical distribution, which pairs with a different local data distribution and a high variance in the numerosity of the available samples.

The main highlight of FEDAVG is the way the model of the server is updated via an iterative procedure; after each client of the federation has trained its model on its own locally available data, a model averaging procedure is carried out by the server, thus updating its model, before propagating the latter to the clients. Even if this strategy proved to be successful for a large number of scenarios, may be limiting for the ones where a limited number of entities, with no computation or communication constraints, constitute the federation; in such a scenario, better performances could be achieved by removing the communication-efficiency related features of the algorithm, thus realizing also complete decentralization. The removal of the centralized server has been vastly investigated; in [23] a peer-to-peer gossip algorithm was proposed to exchange portions of the model weights among computing nodes with limited bandwidth and improve the overall communication efficiency of the federation, while in [24] an algorithm based on a partial aggregation of the gradients produced by the training nodes is proposed.

The present work designs a decentralized version of FEDAVG that easily allows for the integration of any of its recent privacy/communication improvements and demonstrates it on a test scenario in power systems.

III. DECENTRALIZED FEDERATED LEARNING (DECFEDAVG)

Let I be the set of N clients. Given client $i \in I$, let w_i be the vector of trainable parameters/weights of its model, and let $D_i = \{(\alpha_n, \beta_n)\}_{n\in\{1,2,\dots,|D_i|\}}$ be the dataset containing its available input-output pairs, with cardinality $|D_i|$; the total available data is denoted as $\mathbb{D} = \bigcup_i D_i$. Suppose that the clients share the same model architecture, hence the cardinality of the weight vectors is the same, i.e., $|w_i| = |w_j|$ $\forall i, j \in I$.

In the federation, each client i is trained to minimize the loss function $l_i\big((\alpha_n, \beta_n)|w_i\big)$ over its entire dataset D_i: given the generic input α_n, the loss function is used to quantify the quality of the model, with parameters w_i, against the corresponding ground truth value β_n, with $(\alpha_n, \beta_n) \in D_i$. The choice of the loss function depends on the particular machine learning problem to be addressed; in general, regression tasks require the mean squared error, while classification ones the categorical cross-entropy. We set:

$$L_i(w_i) = \frac{1}{|D_i|} \sum_{(\alpha_n,\beta_n)\in D_i} l_i((\alpha_n, \beta_n)|w_i) \tag{1}$$

as *loss function* of client i over its entire dataset D_i. The goal of the federation is then to find the optimal vector w^* of parameters that, when shared by all clients, solves the minimization problem with joint cost function defined as [13]:

$$\min_w \mathcal{L}(w) := \sum_{i\in I} p_i L_i(w) \tag{2}$$

with $p_i = |D_i|/|\mathbb{D}|$. While in the standard machine learning setting optimization (2) is tackled by a centralized system, which computes the gradient $\nabla L(w)$ given the whole set $\mathbb{D}$, in a distributed one, the gradient $\nabla L(w)$ has to be estimated starting from the gradients of the clients $\nabla L_i(w_i)$.

Moreover, in standard (non-federated) distributed learning, data can be distributed arbitrarily by a centralized entity over the clients. The typical assumption for this distribution is that the datasets D_i are IID with respect to $\mathbb{D}$, implying $\mathbb{E}\left[L_i(w)\right] = \mathcal{L}(w)$. In practice, under this assumption $L_i(w)$ provides a good approximation of $\mathcal{L}(w)$ [1] and the locally computed gradients $\nabla L_i(w_i)$ can be averaged to reconstruct $\nabla L(w)$.

On the contrary, in the federated setting such IID hypothesis can not be assumed, as the training data is processed without any re-distribution and $L_i(w)$ could provide an arbitrarily bad approximation of $\mathcal{L}(w)$. For this reason, in FEDAVG [1], [3] the author proposed a round-based iterative procedure for model averaging.

FEDAVG is divided into two main phases, which are repeated iteratively. In the first phase (*local training*), the server selects a subset of clients that update the weights of

Algorithm 1 FedAvg

```
1:  SERVER UPDATE:
2:  for each communication round $t = 1, ..., T$ do
3:      select a subset of clients for the averaging procedure
4:      for all selected client $i$ do
5:          CLIENT UPDATE
6:          receive $\tilde{w}_i$ from client $i$
7:      end for
8:      set $w(t) = \sum_i p_i \tilde{w}_i(t)$
9:      propagate $w$ in the federation $(w_i(t) = w(t), \forall i)$
10: end for

11: CLIENT UPDATE:
12: for each local epoch $e = 1, ..., E$ do
13:     for each mini-batch $b$ from $D_i$ do
14:         $w_i(t-1) \leftarrow w_i(t-1) - \eta \nabla L_i(b|w(t-1))$
15:     end for
16: end for
17: set $\tilde{w}_i(t) = w_i(t-1)$
18: return $\tilde{w}_i(t)$ to the server
```

their models by training on their local datasets D_i with a gradient descent update rule:

$$\tilde{w}_i(t) = w_i(t-1) - \eta \nabla L_i\big(w_i(t-1)\big) \tag{3}$$

where $0 < \eta < 1$ is the learning rate and $\tilde{w}_i(t)$ is the locally updated weight of the model of agent i at time-step t. We mention that in the FL setting it is typically assumed that all clients share a common initial weight vector, i.e., $w_i(0) = w_0$ $\forall i \in I$ [1]. Actually, the local weight update is performed iteratively over a given number of *training epochs* using a variation of gradient descent (*mini-batch gradient descent*) that splits D_i's into a set of *mini-batches*. For the sake of simplicity, equation (3) exemplifies the update rule with $E = 1$ and over the complete dataset, whereas the pseudo-code presented below reports the mini-batch multi-epoch version of the algorithm.

In the second phase (*centralized averaging*), the server collects the $\tilde{w}_i$'s, computes the weight vector $w(t)$ as the weighted average

$$w(t) = \sum_{i \in I} p_i \tilde{w}_i(t) \tag{4}$$

and propagates the weight vector $w(t)$ to all the clients:

$$w_i(t) = w(t), \forall i \in I. \tag{5}$$

We report the pseudo-code for FedAvg (see Algorithm 1), showing an implementation where the clients perform E local training epochs using mini-batch Gradient Descent with a batch size of *B*. In the code, $\nabla L_i\big(b|w_i\big)$ denotes the gradient performed over the mini-batch b and it is assumed for simplicity that all clients participate in the averaging procedure.

Algorithm 2 DecFedAvg

```
1:  DECENTRALIZED FEDERATED TRAINING:
2:  for all communication rounds $t = 1, ..., T$ do
3:      for all clients $i \in I$ do
4:          for each local epoch $e = 1, ..., E$ do
5:              for each mini-batch $b$ from $D_i$ do
6:                  $w_i(t-1) \leftarrow w_i(t-1) - \eta \nabla L_i(b|w(t-1))$
7:              end for
8:          end for
9:          set $\tilde{s}_i(t) = w_i(t-1)$
10:         update $w_i(t)$ according to (6)
11:     end for
12: end for
```

Although several variants of FedAvg have been developed, [13], [25] for instance, the centralized setting and the two-phase approach are the two main shared features.

When communication costs are negligible, the presence of point-to-point communications among the clients and the unavailability of the centralized server are the two main communication constraints. Therefore, the development of a decentralized version of FedAvg, i.e., (3)-(5), comes in handy. A direct decentralization of FedAvg would consist in using equation (3) and the following equation:

$$w_i(t) = \frac{1}{|D_i|}\left(|D_i|\tilde{w}_i(t) + \sum_{j \in N_i} |D_j|\tilde{w}_j(t)\right), \forall i \in I, \tag{6}$$

with $|\mathbb{D}_i| = |D_i| + \sum_{j \in N_i} |D_j|$, where N_i is the set of the neighbours of client i, namely trusted clients of client i with which client i communicates. If all the clients were neighbours of each other, substituting (4) into (5) would yield that (3), (6) are equivalent to (3)-(5). Hereafter, we only assume that the underlying communication graph is connected.

Equation (6) states that, at every communication round t, the clients exchange the weights $\tilde{w}_i(t)$ of their locally trained model with their neighbours. Each client then updates the vector $w_i(t)$ by computing a weighted average of the collected vectors $\tilde{w}_j(t)$, with $j \in N_i \cup \{i\}$ (including its own vector), with weights set as the cardinality $|D_j|$ of the clients' data. In general, arbitrary weights can be attributed to the clients, e.g., depending on the in/out-degree of the nodes in the federation graph or reflecting the trust level that client i has in client j.

The pseudo-code for DecFedAvg is reported (see Algorithm 2) to improve the clarity of the presentation.

Remark 1. The proposed DecFedAvg algorithm does not utilize any information on the communication network topology. Also, we note that the proposed distribution of (3)-(5), for its simplicity, can in principle be applied to any FL algorithm with the same structure as FedAvg. However, the convergence results of such algorithm typically rely on the propagation of a common averaged model into the federation.

Hence, the convergence of the decentralized versions is not guaranteed, as, in general, at each communication round t the various $w_i(t)$ are different and consequently the performance of their clients may differ.

IV. Dataset for NILM

The proposed approach is validated against the original FedAvg over the NILM problem, a technique to recover source appliances from only the recorded mains in a household [26]. It constitutes the preferred approaches to tackle Appliance Load Monitoring (ALM); its non-intrusivity nature makes it easy-to-use in existing buildings, without any inconvenience to householders. Data is collected from sensors installed in buildings which provide active power reading, making it a crucial step towards better energy management.

In our experiments, we used the REFIT [27] dataset; it consists of power readings, sampled every 8 seconds, coming from 20 households in Loughborough, England (UK) from 2013 to 2015. The dataset provides data regarding five appliances for energy disaggregation purposes: kettle, microwave, fridge, dish washer, washing machine. Data is preprocessed by calculating the standard score of each sample, as in [26], and is distributed in the following way: for each appliance, one house is chosen to be the server, while all the others are set to be the federated clients, as shown in Table I.

In this paper, we apply the *sequence-to-point* (*seq2point*) model architecture [28]. Even if there are several state-of-the-art NILM algorithms, which are based on different approaches like Hidden Markov Models (HMM) [29]–[32], Graph Signal Processing (GPS) [33], [34], deep learning [35], [36], *seq2point* outperforms both HMM and deep learning-based approaches. Differently from the *sequence-to-sequence* (*seq2seq*) approach [35], which takes a sequence of items as input and outputs another sequence of items, the *seq2point* learning technique consists in training a model to predict the midpoint element of an output window of appliance readings, which is assumed to be a non-linear regression of the input window (the interested reader may find more details in [28]). The *seq2point* learning technique is applied to the entirety of the federation. As described in the next Section, the *seq2point* approach consists in a sliding window approach which returns sample vectors of size $W = 599$. The number of samples for each client and the server has been set equal to $(1000, 2500, 7500, 1000, 5000, 4000, 9000, 3000, 6400, 1000, 7000)$ and (10000), respectively.

Clients	*dish washer*	*fridge*	*kettle*	*microwave*	*washing machine*
client1	5	2	3	10	2
client2	7	5	4	12	5
client3	9	9	5	17	7
client4	13	12	6	19	9
client5	16	-	7	-	15
client6	18	-	8	-	16
client7	-	-	9	-	17
client8	-	-	12	-	18
client9	-	-	13	-	18
client10	-	-	19	-	-
client11	-	-	20	-	-
server	20	15	2	4	8

TABLE I: Houses-to-Clients correspondence for appliances

V. Test Results

This section reports the test simulations performed to validate the proposed approach. The goal of NILM – or energy disaggregation approach [37] – is to measure the energy consumption of individual appliances based on the total electricity consumption [28].

Given a household, at each sample time k the total power consumption reading in Watts, denoted by $y_k \in \mathbb{R}_+$, $k = 1, ..., K$, is observed, where K is number of considered sample times. The unknown reading of the power consumption of appliance i is denoted by $x_{ik} \in \mathbb{R}_+$, $i = 1, ..., I$, is observed, where I is total number of appliances. The measurements of the mains and of the appliances are collected in two vectors, $Y = \left[y_1, y_2, ..., y_K\right]$, $X_i = \left[x_{i1}, x_{i2}, ..., x_{iK}\right]$.

At each time step k, y_k is interpreted as the sum of the real readings of the appliances of interest, a Gaussian noise factor ϵ with zero mean and variance σ^2, and an unknown factor u_k which represents the fact we are interested in a subset $A \subseteq I$ of all the possible appliances: $y_k = \sum_{i=1}^{A} x_{ik} + u_k + \epsilon$.

By applying the *seq2point* approach with window size W, let $Y_k^W = \left[y_k, ..., y_{k+W-1}\right]$ be th k-th input, with $k \leq K - W + 1$, let $x_{i\tau}$ be the middle point of the corresponding appliance window $X_{ik}^W = \left[x_{ik}, ..., x_{i(k+W-1)}\right]$, with $\tau = k + \left\lceil \frac{W}{2} \right\rceil$, and let $\hat{f}_i$ be the function approximated by the model, yielding the output

$$\hat{x}_{i\tau} = \hat{f}_i\left(Y_k^W\right), \tag{7}$$

where $\hat{x}_{i\tau}$ is the model prediction for the considered middle point for appliance i.

We tested our algorithm on each appliance data, kettle (11 clients), microwave (4 clients), fridge (4 clients), dishwasher (6 clients), washingmachine (8 clients), as shown in Table I, comparing its performance with that of FedAvg in terms of the mean absolute error (MAE), as shown in Table II.

A. Test Setup

The model architecture used for *seq2point* learning consists of an input of dimension $W = 599$, 5 convolutional layers of dimensions $(30, 10, 1)$, $(30, 8, 1)$, $(40, 6, 1)$, $(50, 5, 1)$, $(50, 5, 1)$ respectively, followed by a dense layer of 1024 units, all with a ReLu [38] activation function, and an output layer of 1 unit, as depicted in Figure 2.

The optimizer chosen for training was ADAM Optimizer [39], with learning rate 0.001, beta 1 0.9, beta 2 0.999, and epsilon 10^{-8}. For both DecFedAvg and FedAvg, we set $E = 3$ (number of epochs in the clients' update), $B = 64$ (local batch size), $R = 20$ (number of communication rounds). In particular, for DecFedAvg, due to the absence of the centralized server, a circle topology was chosen, as depicted in Figure 3.

We recall that the simulations were done using Tensorflow and Keras. All clients used the same architecture depicted in Figure 2 for all the testings.

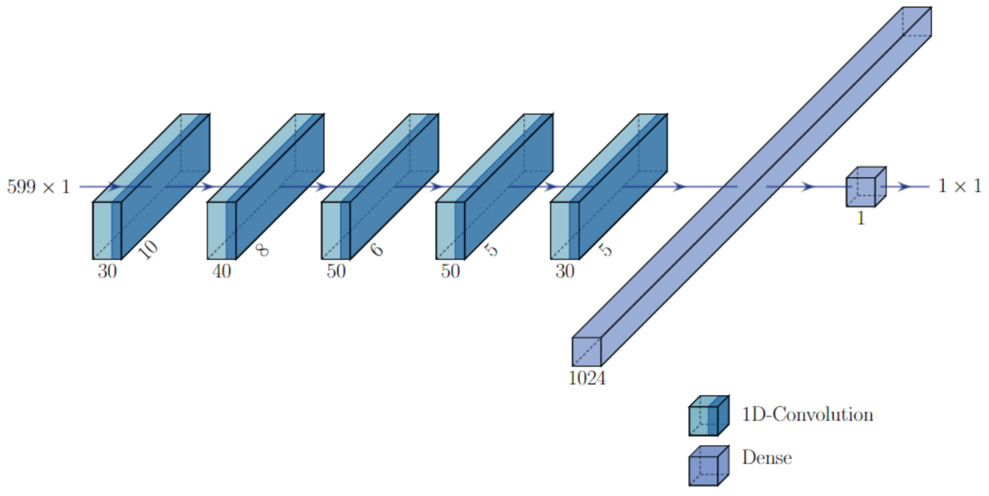

Fig. 2: Neural Netowrk Architecture, as in [26]

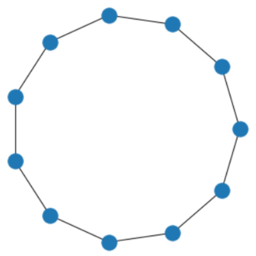

Fig. 3: Circle federation topology, kettle

Appliance	**MAE FedAvg**	**MAE DecFedAvg**	**Loss % Error**	**MAE % Error**
dish washer	0.0708	0.0747	5.46%	5.50%
fridge	0.0915	0.0916	0.030%	0.032%
kettle	0.0290	0.0271	0.87%	6.42%
microwave	0.0350	0.0255	10.37%	16.30%
washing machine	0.0588	0.0553	0.15%	5.96%
average	0.05702	0.0548	3.38%	6.84%

TABLE II: FEDAVG vs DECFEDAVG performance comparison

B. Simulations

DECFEDAVG is tested against FEDAVG over each appliance, considering the same clients and corresponding data distribution. Being a regression problem, the performance metric used for a comparison is the mean absolute error (MAE), evaluated with respect to the ground truth of clients data. Moreover, for the sake of a fair comparison of the two algorithms, we evaluated also the loss and MAE relative error. Table II shows our results.

Figure 4 shows the comparison of the MAE evolution for appliance washing machine (the DecFedAvg line is the one observed by the first client). It can be noted how similar the behaviour of the two approaches are, despite the absence of a centralized entity, with DECFEDAVG even managing to attain lower MAE values over the communication rounds. Similar behaviours are observed for the other appliances and hence their graphs are omitted.

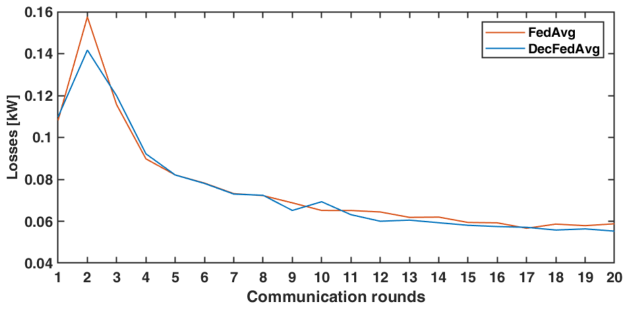

Fig. 4: FedAvg vs DecFedAvg: MAE comparison for washing machine

VI. CONCLUSIONS AND FUTURE WORKS

This paper presents a novel decentralized Federated Learning algorithm, DECFEDAVG, obtained as a direct decentralization of the original Federated Learning algorithm, FEDAVG.

On going and future work is aimed at using the proposed approach to tailor solutions to specific use cases, and to modify the algorithm on the basis of consensus theory to achieve exact decentralization for arbitrary communication topologies.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017.

[2] 2018 reform of eu data protection rules. European Commission. [Online]. Available: https://eur-lex.europa.eu/legal-content/IT/TXT/?uri=celex%3A32016R0679

[3] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *ArXiv*, vol. abs/1602.05629, 2016.

[4] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[5] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for uav-enabled networks: Learning-based joint scheduling and resource management," 2020.

[6] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2020.

[7] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International Journal of Medical Informatics*, vol. 112, pp. 59–67, Apr. 2018. [Online]. Available: https://doi.org/10.1016/j.ijmedinf.2018.01.007

[8] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," 2018.

[9] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, "Edgefed: Optimized federated learning based on edge computing," *IEEE Access*, vol. 8, pp. 209 191–209 198, 2020.

[10] L. U. Khan, M. Alsenwi, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "Resource optimized federated learning-enabled cognitive internet of things for smart industries," *IEEE Access*, vol. 8, pp. 168 854–168 864, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.3023940

[11] H. Liu, X. Zhang, X. Shen, and H. Sun, "A federated learning framework for smart grids: Securing power traces in collaborative learning," 2021.

[12] H. Wang, C. Si, and J. Zhao, "A federated learning framework for non-intrusive load monitoring," 2021.

[13] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, p. 1–1, 2021. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2021.3124599

[14] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[15] A. Giuseppi, L. D. Torre, D. Menegatti, and A. Pietrabissa, "AdaFed: Performance-based adaptive federated learning," in *2021 The 5th International Conference on Advances in Artificial Intelligence (ICAAI)*. ACM, Nov. 2021. [Online]. Available: https://doi.org/10.1145/3505711.3505717

[16] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2017. [Online]. Available: https://doi.org/10.1145/3133956.3133982

[17] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2018.

[18] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," 2019.

[19] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 2512–2520.

[20] S. Kim, "Incentive design and differential privacy based federated learning: A mechanism design perspective," *IEEE Access*, vol. 8, pp. 187 317–187 325, 2020.

[21] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2017.

[22] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," 2019.

[23] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," 2019.

[24] J. Jiang and L. Hu, "Decentralised federated learning with adaptive partial gradient aggregation," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 230–236, Sep. 2020. [Online]. Available: https://doi.org/10.1049/trit.2020.0082

[25] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020.

[26] M. DIncecco, S. Squartini, and M. Zhong, "Transfer learning for non-intrusive load monitoring," 2019.

[27] D. Murray, L. Stankovic, and V. Stankovic, "An electrical load measurements dataset of united kingdom households from a two-year longitudinal study," *Scientific Data*, vol. 4, no. 1, Jan. 2017. [Online]. Available: https://doi.org/10.1038/sdata.2016.122

[28] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, "Sequence-to-point learning with neural networks for nonintrusive load monitoring," 2017.

[29] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *Proceedings of the 2011 SIAM international conference on data mining*. SIAM, 2011, pp. 747–758.

[30] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, N. D. Lawrence and M. Girolami, Eds., vol. 22. La Palma, Canary Islands: PMLR, 21–23 Apr 2012, pp. 1472–1482. [Online]. Available: https://proceedings.mlr.press/v22/zico12.html

[31] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, ser. AAAI'12. AAAI Press, 2012, p. 356–362.

[32] S. Makonin, F. Popowich, I. Baji ´c, B. Gill, L. Bartram, and I. Bajic, "Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring (nilm)," *IEEE Transactions on Smart Grid*, vol. (in press), pp. 1–11, 10 2015.

[33] B. Zhao, L. Stankovic, and V. Stankovic, "On a training-less solution for non-intrusive appliance load monitoring using graph signal processing," *IEEE Access*, vol. 4, pp. 1784–1799, 2016. [Online]. Available: https://doi.org/10.1109/access.2016.2557460

[34] V. Stanković, J. Liao, and L. Stanković, "A graph-based signal processing approach for low-rate energy disaggregation," *2014 IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES)*, pp. 81–87, 2014.

[35] J. Kelly and W. Knottenbelt, "Neural nilm," *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, Nov 2015. [Online]. Available: http://dx.doi.org/10.1145/2821650.2821672

[36] P. P. M. do Nascimento, "Applications of deep learning techniques on nilm," *Diss. Universidade Federal do Rio de Janeiro*, 2016.

[37] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[38] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018. [Online]. Available: http://arxiv.org/abs/1803.08375

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.