Politecnico di Torino

Politecnico di Torino

Master of Science in Communications and Computer Networks Engineering
A.y. 2022/2023

Master's Thesis

A Survey on Federated Learning Algorithms in IoT networks

Academic Supervisor:
Prof. Claudio Ettore Casetti

Candidate:
Viktoria Muradyan

Contents

A Survey on Federated Learning Algorithms in IoT networks

Abstract

The capability of mobile devices to sense and compute at an advanced level has significantly improved in recent years, and combined with the advancements in Deep Learning (DL) has led to a vast range of opportunities for different practical applications such as healthcare and vehicular networks. However, traditional cloud-based Machine Learning (ML) approaches require the data to be stored in a cloud server, resulting in critical issues such as poor communication efficiency, high latency and privacy concerns. The idea to bring intelligence closer to the edge was proposed. However, the conventional technologies that enable Machine Learning (ML) at edge networks require sharing of personal data, which defies strict data privacy regulations. In response, Federated Learning (FL) has been introduced. In FL, devices use their local data to train a ML model in a decentralized way. Instead of sharing raw data, devices iteratively optimize the model locally. Then, the resulting model updates are shared to a centralized server for aggregation. However, in a complex mobile edge network that includes various devices with different constraints, some challenges may arise. Implementing FL on a large scale presents difficulties with regards to communication costs, resource allocation, as well as privacy and security concerns. To address these challenges, this survey paper provides an overview of the fundamentals and background of FL and its implementation challenges. We focus on reviewing client selection frameworks for FL, then we shed light on state-of-the-art (SoTA) techniques that tackle the issue of data heterogeneity. Lastly, we provide a detailed discussion about novel communication efficienct FL algorithms that strike a balance between learning accuracy and communication efficiency, via model masking and pruning.

1. Introduction

Nowadays there are numerous amount of IoT and mobile devices. Each of these devices produce multitudes of information, which can be beneficial to improve Artificial Intelligence (AI) for different applications. In traditional Centralized Machine Learning (ML) settings, data are pooled from various data sources to train ML models at the cloud-server side. In most applications, data sources correspond to simple, computationally capable, battery limited sensory devices with communication abilities (e.g. IoT devices, phones, etc $\cdots$). Generally, training robust ML models necessitates the availability of large amount of training data. As a result, centralized ML training requires a set of communication resources and an energy budget (i.e. bandwidth and transmission power) exploding with the number of devices in the system and the size of data pooled. Moreover, in some settings, the transmission of raw data to a central server may raise privacy concerns, especially if the data hold sensitive and private user information. To address the aforementioned problems, a set of training algorithms known as Distributed Learning (DL) has been proposed. In DL, devices collaborate to train a model locally, without the need to transmit any raw data samples to the cloud. Collaboration is orchestrated by a central server, or established directly via Device-to-Device (D2D) communication links. In the latter case collaboration is established among neighbouring devices [18] to solve ML problem. D2D is chosen, when server-clients based approaches are undesirable. Distributed Learning encounters the following challenges:

- When devices are connected through wireless links, impairments will be introduced affecting the transmitted data (e.g., interference, noise, fading) which should be accounted for in the system model.
- Collaboration among devices in a distributed fashion may impose learning latency. Consequently, it becomes necessary to strike a balance between distributed model training delay and accuracy in some delay-intolerant application.

There are different DL frameworks proposed in the literature, among the most popular ones is *Federated Learning (FL)*, which will be explained in details in the next section.

2. Federated Learning

In a supervised FL setting, the system is made up of a set $\mathcal{K}$ of K devices (clients) and a server. Each device k is endowed with a dataset $\mathcal{D}_k = \{(\mathbf{x}_{k,i}, y_{k,i})\}_{i=1}^{|\mathcal{D}_k|}$, where, $\mathbf{x}_{k,i} \in \mathbb{R}^d$ denotes a feature input vector, and (without any loss of generality) $y_{k,i} \in \mathbb{R}$ denote its corresponding ground truth label. Supervised by the server, users collaborate to train a ML model to fit their corresponding task. In particular, the goal of collaborative training is to find a global model that can minimize the average loss among the devices given by:

$$F(\mathbf{w}) = \sum_{k=1}^{K} p_k F_k(\mathbf{w}), \qquad \text{s.t} \quad \sum_{k} p_k = 1, \tag{1}$$

where p_k denotes the weight of user k, $F_k(\mathbf{w})$ is the average local loss function of device k over its local dataset, given by:

$$F_k(\mathbf{w}) = \frac{1}{|\mathcal{D}_k|} \sum_{i=1}^{|\mathcal{D}_k|} f(\mathbf{x}_{k,i}, y_{k,i}, \mathbf{w}), \tag{2}$$

where $f(\mathbf{x}_{k,i}, y_{k,i}, \mathbf{w})$ is the local loss of the model $\mathbf{w}$ over the example pair $(\mathbf{x}_i, y_i)$. The FL training goal is to find a minimizer $\mathbf{w}^*$ of (1) such that:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w}), \tag{3}$$

This is achieved via an iterative process, where each device k updates its local model at each iteration $t+1$ according to:

$$\mathbf{w}_k[t+1] = \mathbf{w}[t] - \eta \nabla F_k(\mathbf{w}[t]), \tag{4}$$

where η is the learning rate and ∇ is the gradient operator. Then, users send their updated model (or model updates) to the server. The server aggregates

the updates according to

$$\mathbf{w}_g[t+1] = \sum_{k=1}^{K} p_k \mathbf{w}_k[t] \tag{5}$$

The produced global model is transmitted to the devices and a new iteration of training is initiated. FL process can be summarized as follows:

Initialization: In the beginning of the training the server initializes the task, the requirements for performing the task, and initializes the global model with all necessary parameters (e.g. learning rate) and sends them to the participating devices.

Model training and update: Each client after receiving the global model trains the ML model using any training method (e.g., SGD Stochastic Gradient Descent) on its local data, and then updates and sends the model updates back to the server.

Global aggregation and model update: The server aggregates all model updates from the devices and updates the global model and sends it back to the participating devices.

This iterative process repeats until FL converges or reaches the desired accuracy.

The authors in [8] describe four key metrics to evaluate FL performance over wireless networks:

1. *Training Loss:* The value of the average loss function $F(\mathbf{w})$, which depends on the model performance at all devices.
2. *Convergence Time:* The time when FL converges or reaches the desired accuracy. It depends on the time needed for clients to train and transmit the models to the server, and the total number of communication rounds.
3. *Energy consumption:* The energy consumption at each participating device. It includes the energy spent on training and transmitting the ML model.
4. *Reliability:* It is the probability that the FL model achieves a target training loss.

The architecture and the training process of FL are illustrated in Fig. 1.

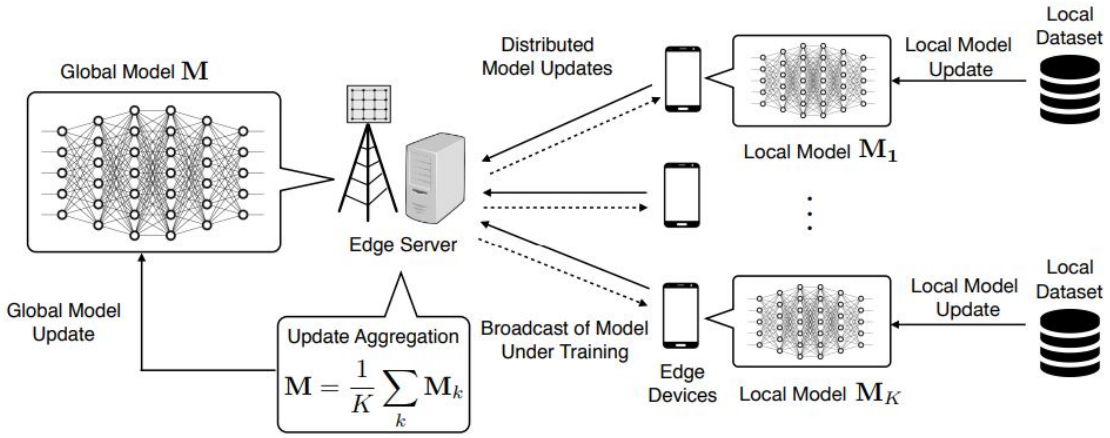


Figure 1: General Federated Learning training process. [23]

2.1 Federated Averaging

Federated Averaging (FedAvg) algorithm was proposed in [14]. It is a training algorithm based on local SGD, which allows users to collaboratively minimize the average loss of the FL model over their local datasets during multiple communication rounds. The FedAvg algorithm procedure is akin to that described in Sec. 2. User weights are assigned according to $p_k = \frac{|\mathcal{D}_k|}{\sum_k |\mathcal{D}_k|}$, i.e the ratio of user k dataset size w.r.t the total number of data available in the system. Moreover, unlike FedSGD which limits the local training to a single step Batch Gradient Descend, FedAvg allows users to apply multiple SGD steps locally over a set of mini-batches before uploading their updates. Consequently, FedAvg is shown to enjoy a faster convergence rate in homogeneous settings, where local users gradients are seen as unbiased estimates to the averaged global gradient calculated at the server side. At the end of each communication round, the produced local models are sent to the server, which in turn aggregates them to produce a new global model. Accordingly, the aggregation rule of FedAvg at communication round $t+1$ is given by

$$\mathbf{w}_g[t+1] = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{\sum_k |\mathcal{D}_k|} \mathbf{w}_k[t], \tag{6}$$

where $w_g[t+1]$ denotes the global model produced at communication round $t+1$, and $w_k[t]$ is given in (4). The pseudo-code of FedAvg is given in Algorithm 1.

Algorithm 1: Federated Averaging

Input : K clients are indexed by k, B is the local minibatch size, E is the number of local epochs and η is the learning rate.

Server executes: initializes w_0

for *each round* $t = 1, 2, \cdots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of of m clients)
  for *each client* $k \in S_t$ ***in parallel*** **do**
    $w_k[t+1] \leftarrow$ Client Update $(k, w_g[t])$
    $w_g[t+1] \leftarrow \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{\sum_k |\mathcal{D}_k|} w_k[t+1]$
  end
end

Client Update (k, w)**:** //Run on client k
$\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size B)
for *each local epoch* i *from* 1 *to* E **do**
  for *batch* $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \bigtriangledown f(w; b)$
  end
end
return w to the server

Numerous challenges face the implementation of FL. Albeit that there is no need to transmit users raw data to the server during training, the dimensionality of model updates transmitted are not negligible. Given the limited radio resources, communication cost remains an issue. Another problem which hinders the convergence of FL models is the heterogeneity of the participating device, which can take different forms. For instance, hardware heterogeneity (i.e. battery capacity and computational power variance across clients) may

limit the participation of some devices during the entire FL training, and introduces stragglers in the system. Additionally, data heterogeneity across devices may lead the FL model to diverge, especially in cases where clients aim to train conflicting tasks. In [21], the authors demonstrate how data heterogeneity inflicts a hit to the training performance of FL. They propose to have the server create a small dataset by collecting few samples from each user dataset. Then, the dataset is shared across the devices to alter their non-i.i.d (independent and identically distributed) local data distribution towards that of the global system distribution. However, pooling data from users to create such a dataset defies the privacy considerations of the FL framework.

2.2 Communication Bottleneck in FL system

Given that the communication between the server and the devices is done through wireless links, the wireless factors affecting the model updates should be considered. Table 1 illustrates the causality among wireless and system variables, and FL performance metrics. For instance, a poor wireless channel can cause errors in the transmission of model updates, leading to stagnation in training loss. Moreover, this would deteriorate the channel capacity, increasing transmission delays for model updates or, if compensated for, higher transmission power requirements.

Table 1: Effects of Wireless factors on FL metrics. A tick implies that a given wireless factor affects the FL metric [8]

Communication Factors	**Training Loss**	**Local Training Time**	**FL Parameter Transmission Time**	**Total Number of Learning Steps**	**Energy Used for Local Training**	**Energy Used for FL Transmission**	**Reliability**
Spectrum resource	√		√	√		√	√
Computational capacity	√	√		√	√		
Transmit power	√		√	√		√	√
Wireless channel	√		√	√		√	√
Set of devices that participate in FL	√		√	√			√
Size of the FL parameters trained by each device	√	√		√	√		√
Size of the FL parameters transmitted by each device			√			√	

Communication bottleneck is another issue while transmitting large amount of data from a substantial number of devices to the server. In FL, due to the

high dimensionality of model updates, it is still very costly to send them to the server. Some proposed solutions to reduce the amount of information exchanged between participating devices and the server through sparsification and quantization [4].

Sparsification transforms an N dimensional vector (i.e. the model update), into a sparse representation, by setting some of its elements to zero according to a pre-defined rule. For instance, the authors in [2], [3], [17] utilize the top-K sparsification rule to sparsify the model updates of the FL devices before sending them to the server. The top-K sparsification rule sets all but the largest absolute values of a vector to zero. This operation rule can be given as:

$$\tilde{\mathbf{w}} = \mathbf{M} \otimes \mathbf{w}, \tag{7}$$

where $\otimes$ is the Hadamard product, $\mathbf{w}$ is the model update and $\tilde{\mathbf{w}}$ denotes its' sparse representation. $\mathbf{M}$ is an N dimensional mask vector $\mathbf{M} \in \{0,1\}^N$.

Devices then proceed to send the non-zero values to the server, alongside indicating the location of non-zero elements to enable the correct reconstruction of sparse model update. This causes an additional communication cost as a function of the number of non-zero elements. As a rule of thumb, the lower is the sparsification level (the ratio of non zero elements in a vector to its dimension) the higher is the compression, and the smaller will be the communication cost of sending the location of non-zero elements.

Since the values of model updates are real, *Quantization* is performed to restrict those values to a finite discrete codebook values to further reduce the number of bits needed to transmit the data.

Despite that sparsification and quantization guarantee a communication efficient transmission of updates, it comes at the price of adding noise to the model updates. As a result, the global model may generalize poorly. Accordingly, it is of vital importance to strike a balance between the communication cost that can be tolerated and the required learning performance depending on the requirements of the task being trained.

2.3 Over the Air computation

Generally, the wireless resources in a FL system increase linearly with the increasing number of devices. This is especially true in orthogonal access schemes such as OFDMA, where the bandwidth is split into sub-channels which are distributed among the users. One problem faced by adopting such a transmission scheme in FL uplink, is that it doesn't scale well with the number of devices in the system, given a finite bandwidth. To tackle this problem, the authors in [24] propose Over the Air computation (AirComp), which leverages the superposition property of radio waves to sum-up and average uncoded analog modulated model updates sent by the users. Their solution stems from the fact that the server is interested in the average model update to produce the global model, rather than each of them individually. In particular, the authors propose to replace digital modulation with linear analog modulation and channel inversion power control at the receiver side, assuming that the channel state information (CSI) are known. Model updates are amplitude modulated into symbols and divided into blocks, such that each block is transmitted over one frequency sub-channel using OFDMA technique. All users share the sub-channels and the transmitted signals are super imposed over the air. Subsequently, an aggregated signal is received at the server, thanks to the interference of the radio waves. As a result, transmission rates and latency become dependent on the total bandwidth rather than the number of participants. Sub-channels are inverted by power control, so the model updates transmitted will be received with the same amplitude, which is one of the requirements to implement over the air computation. Due to deep fading, truncated channel inversion is used, where each user's sub-channel is inverted only once its gain exceeds a given threshold. Accordingly, some users' model updates coefficients may be lost, which would negatively affect the FL accuracy and convergence time. Another important requirement for implementing the AirComp is time synchronization of participating devices, which can be done by adopting *timing advance* techniques. Fig. 2. illustrates AirComp procedures.

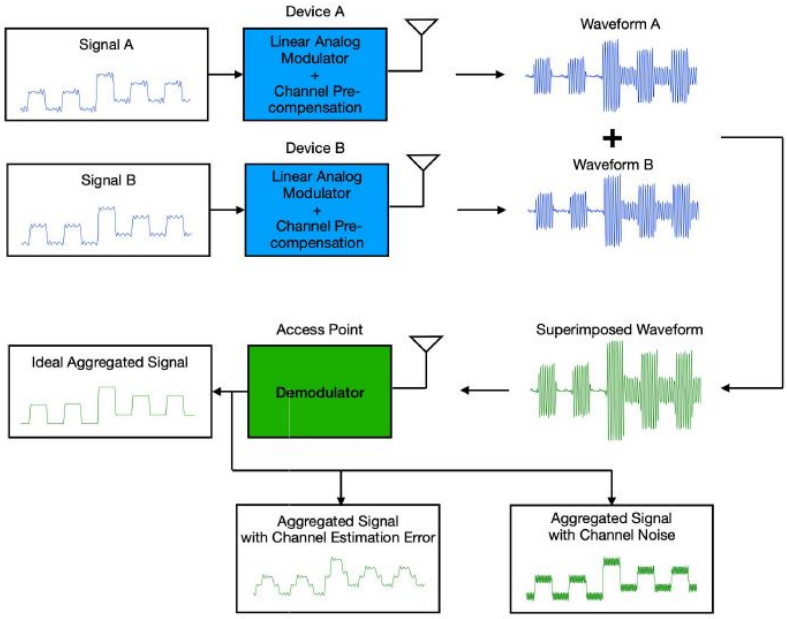
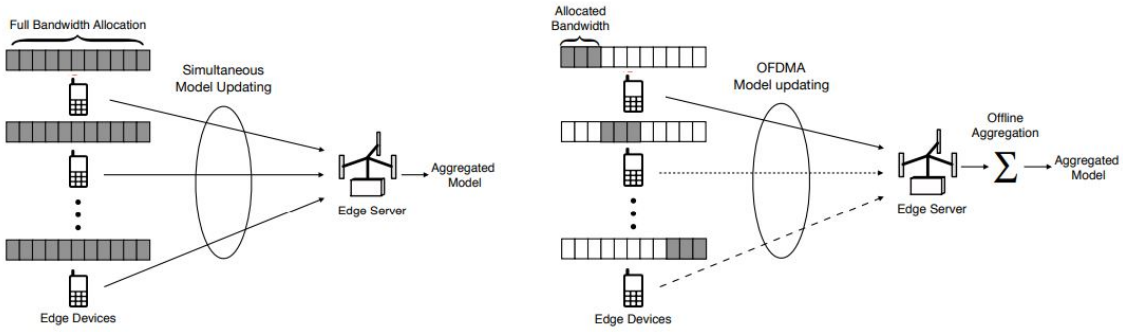


Figure 2: Demonstration of over the air model aggregation (AirComp) [24]

Fig. 3 illustrates the difference between AirComp and OFDM transmissions in FL.



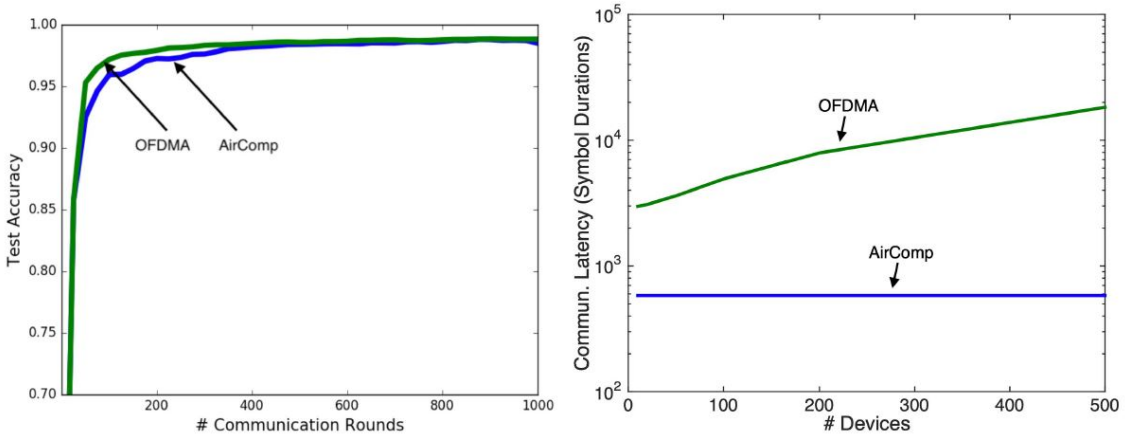
(a) Broadband analog aggregation by AirComp

(b) Broadband digital aggregation by OFDMA

Figure 3: Difference between AirComp and Digital OFDMA techniques [23]

One problem that faces AirComp relates to device scheduling. Specifically, devices residing far away from the server may experience deep fading. Accordingly, distant devices are ignored and only close devices are scheduled.

However, this may impact the testing accuracy, as data diversity could be lost. In FL, it is of utmost importance to guarantee data diversity among the devices, to avoid training a biased model.



(a) Test accuracy (b) Communication latency

Figure 4: Performance comparison between AirComp and OFDMA [8]

Fig. 4 highlights the performance difference between digital OFDMA and AirComp in terms of test accuracy and latency as a function of the number of communication rounds and number of devices, respectively. We note that the FL model under AirComp converges slower compared to digital OFDMA, due to the errors that follow from the channel truncation step in the uplink. Moreover, according to Fig. 4b, AirComp is shown to significantly reduce the communication latency by a factor proportional to the number of devices involved in the process in comparison with the digital OFDMA approach, while guaranteeing a comparable learning accuracy.

2.4 A Joint Communication and Learning Framework

In [7] a novel framework is proposed for implementation of FL over the wireless links, by jointly taking into account FL convergence and wireless effects. In this framework, participating devices are cellular users. For uplink transmissions

OFDMA technique is used and each user is given one resource block (RB). Moreover, the authors assume that the model updates are sent to the Base Station (BS) in a single packet and that erroneous packets (i.e model updates) are discarded by the server. Then, the users packet error rates are quantified as a function of their transmission power and resource allocation. Furthermore, an upper bound of the average excess risk is derived as a function of the user selection scheme, and their average packet error rates. The upper bound of the average excess risk is given by:

$$\mathbb{E}\left(F(\mathbf{w}_{t+1}) - F(\mathbf{w}^*)\right) \leq A^t \mathbb{E}\left(F(\mathbf{w}_0) - F(\mathbf{w}^*)\right) + \underbrace{\frac{2\zeta_1}{LK} \sum_{i=1}^{U} K_i \left(1 - a_i + a_i q_i\left(\mathbf{r}_i, P_i\right)\right) \frac{1 - A^t}{1 - A}}_{\text{Impact of wireless factors on FL convergence}}, \quad (8)$$

where $F(\mathbf{w}_{t+1})$ is the average loss of the global model (over the devices datasets) at round $t+1$, $\mathbf{w}_0$ is the initialized model, and $F(\mathbf{w}^*)$ is the global minimum of F (under the assumption that F is convex) in the ideal setting where wireless impairments are absent, and all users successfully participate in training. The excess risk, quantified by the gap between the average and the optimal loss at round $t+1$ is mainly regarded to the wireless factors introduced in the system. a_i is a binary variable indicating the scheduling of user i, $\mathbf{r}_i$ is a one-hot encoded vector associating user i to a resource block, P_i is user's i transmission power, and A is the convergence rate, which is a function of the aforementioned parameters. Finally, the authors aim at minimizing the average excess risk bound by jointly optimizing the scheduling of users, their resource allocation and transmission power, alongside bounding the convergence rate below one.

3. Client Selection in Federated learning

Given the limited wireless resources available in the FL system, it becomes essential to determine which clients shall be scheduled and chosen to participate

in training. Generally, the server in the FL system waits for all the clients to upload their models before the aggregation takes place. Therefore, devices with low computational capabilities or bad channel conditions may introduce improper delays that impact the convergence time. One of the solutions is setting a deadline for the UL step and discard all model updates received after. However this approach may lead to wasting the wireless and clients resources, as users are required to run the global model optimization locally and are often granted a wireless resource to transmit their updates once done. A better solution is to perform clients selection prior or during training, to induce an efficient use of resources and some performance guarantees. We now shed light on some client selection frameworks in the literature.

3.1 Federated Learning with Client Selection

Federated Learning with Client Selection (FedCS) framework is proposed in [15]. FedCS consists of a Mobile edge Computing (MEC) platform (base station and a server) and the clients. The difference between FedCS and original FL, is that after the initialization step, the server sends a resource information request to a random set of selected clients which constitute their wireless channel condition, computational capabilities and the amount of data they posses.
After receiving the resource information, the MEC operator decides on which clients will participate in the training, and then broadcasts the global model to the selected clients. In Fig.5 FedCS steps are shown.
According to [14] the large number of clients participating in FL training speeds up the convergence time, hence it is important to maximize the number of clients in the training process. Accordingly, the authors in [15] devise the following client selection optimization problem

$$\max_{\mathbb{S}} \quad |\mathbb{S}| \tag{9}$$

$$\text{s.t. } T_{round} \geq T_{CS} + T_{\mathbb{S}}^{d} + \Theta_{|\mathbb{S}|} + T_{agg} \tag{10}$$

where $\mathbb{S}$ set of selected clients for the training process, T_{round} denotes a fixed round duration, T_{CS} is the required time for client selection, $T_{\mathbb{S}}^{d}$ represents

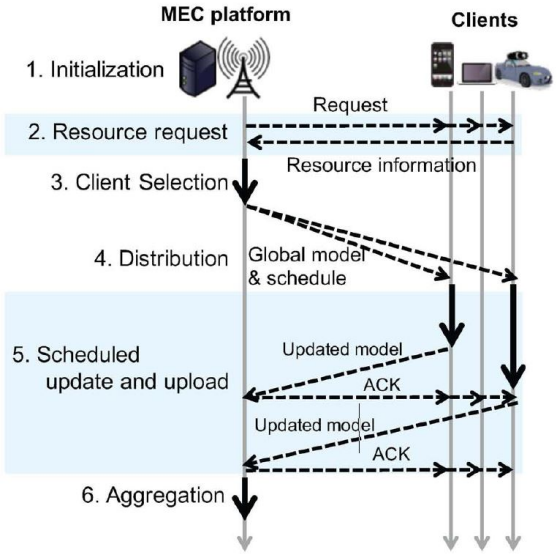


Figure 5: FedCS framework. [15]

the required time for the global model distribution, T_{agg} the aggregation time and $\Theta_{\mathbb{S}}$ is the time needed by the slowest client in the selected set to update and upload its updates to the server. Another important parameter is T_{final}, which denotes the deadline for FL training. Accordingly, the number of communication rounds can be given as $\frac{T_{final}}{T_{round}}$. Note that if T_{round} is increased, the number of clients able to participate in training process will subsequently increase, but the total number of communication rounds before the final deadline will decrease.

In Figures 6 and 7, the performance of FedCS in i.i.d and non-i.i.d settings, compared to FedLim is highlighted respectively. FedLim represents the original FL equipped with a deadline for each communication round, where late updates are discarded once the deadline is exceeded. Thanks to the optimization problem solved in (10), it is shown that FedCS achieves higher accuracy than FedLim in both settings, given the same communication round deadline. Those results stem from the fact the FedCS is able to efficiently accommodate higher client participation rates and guarantee better performance results regardless of the induced data heterogeneity.

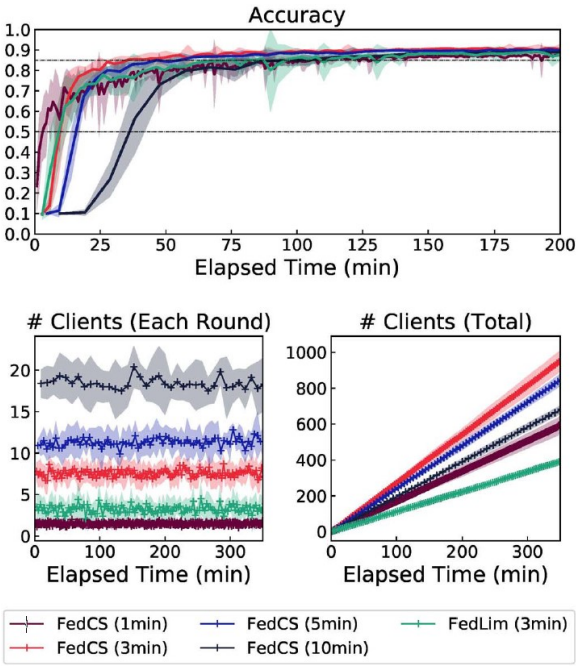


Figure 6: FedCS performance in i.i.d setting [15]

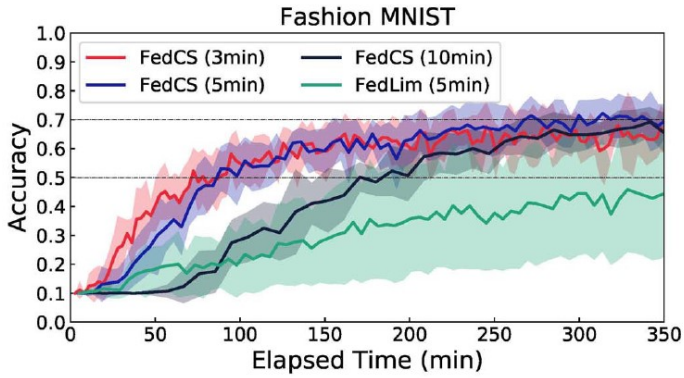


Figure 7: FedCS performance in non i.i.d setting [15]

3.2 Multi Criteria Client Selection

In [5], it is reported that the number of participating clients is 4× more during the night, while they are idle. As a consequence, the location of the clients will impact their participation in training. In [1], which is the case study for an intrusion detection system, a new client selection model is proposed,

while considering clients location. Similar to FedCS, their proposed algorithm aims at maximizing the number of participating devices in FL training, while accounting to their resources in addition to their geographical location. The steps of the proposed FL approach can be summarized into multiple steps:

- *Initialization:* Server generates the global model.
- *Client Filtering:* Instead of choosing clients at random, the clients are formed into subgroups with the users sharing similar characteristics. In this proposed algorithm users are filtered according to their locations.
- *Resource and Request:* After filtering the clients, the server request the clients' resource information.
- *Multicriteria Clients Selection:* Once the resource information is received, the server selects the clients able to perform FL training.
- *Distribution:* The server distributes the global model to the selected clients.

The remaining steps are akin to the original FL algorithm.
Denoting $X = (X_1, \cdots, X_K)$ the set of all clients and $X_f = (X_{f1}), \cdots, X_{fK})$ the filtered client set. In accordance with [15], the goal is to maximize the number of participating clients under the following constrains:

$$\max_{S} \quad |S| \tag{11}$$

$$s.t. \quad \forall X_{\{f_z\}_{z=1}^{i}}, \quad \sum Util^{X_{fz}}_{r\in(CPU,Memory,Energy)} < Budget_r^{X^{fz}}, \tag{12}$$

$$\forall X_{\{f_z\}_{z=1}^{i}}, \quad \sum (T_d^{X_{fz}} + Util^{X_{fz}}_{r=T_{ud}} + T_{ul}^{X_{fz}}) < T, \tag{13}$$

$$s.t. \quad max \quad ER_{X_{\{f_z\}_{z=1}^{i}}} = \left[\frac{|X_{f_z} \cdot l_A|}{|X_{f_z} \cdot l_A| + |X_{f_z} \cdot l_N|} \times 100 \right]. \tag{14}$$

The first constrain indicates that the utilized resources must not exceed a limit budget, which is dynamic per device type. $Util^{X_{fz}}_{r\in(CPU,Memory,Energy)}$ are the predicted utilized resources using linear regression. The second constrain indicates that the download, upload and update time should not exceed a given threshold, where $Util^{X_{fz}}_{r=T_{ud}}$ is the predicted update time (i.e. the local update time). The third constrain indicates that the client selection depends on the Event Rate (ER) of each client's data set, showing the class distribution of

minor samples. It aims at guaranteeing a representative unbiased model, by accounting to the datasets of the selected clients. Accordingly, the clients with high ER are prioritized and chosen to participate in the training. In Fig. 8

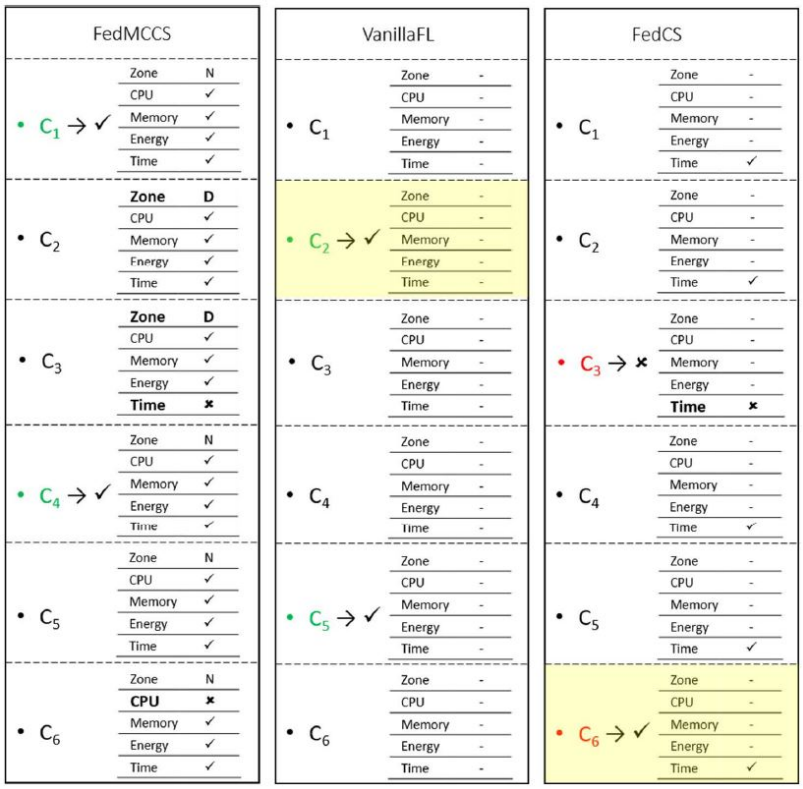


Figure 8: The comparison of client selection algorithm,when $k = 2$ [5]

shows a comparison of three techniques when 2 clients need to be selected. VanillaFL is the original FL, which randomly chooses clients C_2 and C_5, while disregarding their resources. As a result, they may not be able to successfully perform the training task. FedMCCS first filters the clients according to their time zone (Day and Night times), then chooses the clients which satisfy the constraints in (12-14). In this case, clients C_1 and C_4 are selected. On the other hand, FedCS randomly chooses C_5 and C_6, while disregarding their time zones. After receiving their resource information, only C_6 will participate in the training since it satisfies the latency requirements needed. As a result, fewer devices are selected to participate in training, and lower convergence rates are

expected. This highlights the importance of accounting to the devices time-zones in the selection process, as devices are often idle at night, and therefore posses enough energy resources to take part in training.

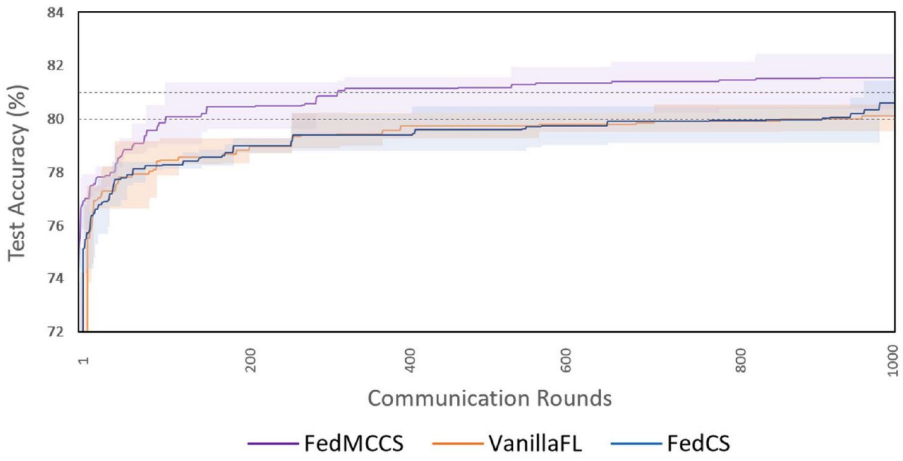


Figure 9: Performance in terms of accuracy among FedMCCS and other baselines [5]

In Fig. 9, the test accuracy attained vs communication rounds for the three algorithms is shown. FedMCCS is shown to converge faster and attains higher accuracy compared to Vanilla FL and FedCS, as a result of the larger participation rate that it guarantees. In Fig. 10 a) illustrates the number of chosen devices for training process and b) shows the actual number of clients performing the training task. It is obvious FedCS and Vanilla FL suffer from more dropouts compared to FedMCCS during most rounds.

3.3 Client Selection for Federated Learning With non-i.i.d Data

A new algorithm for client selection is proposed in [20], called CSFedAvg, where *weight divergence* is used to differentiate between different non-i.i.d degrees of the clients' data. Their goal is to choose clients to reach an unbiased model, representative of the population target data distribution. To this end, The proposed approach uses a virtual client C_0 at the server with an i.i.d dataset (i.e.

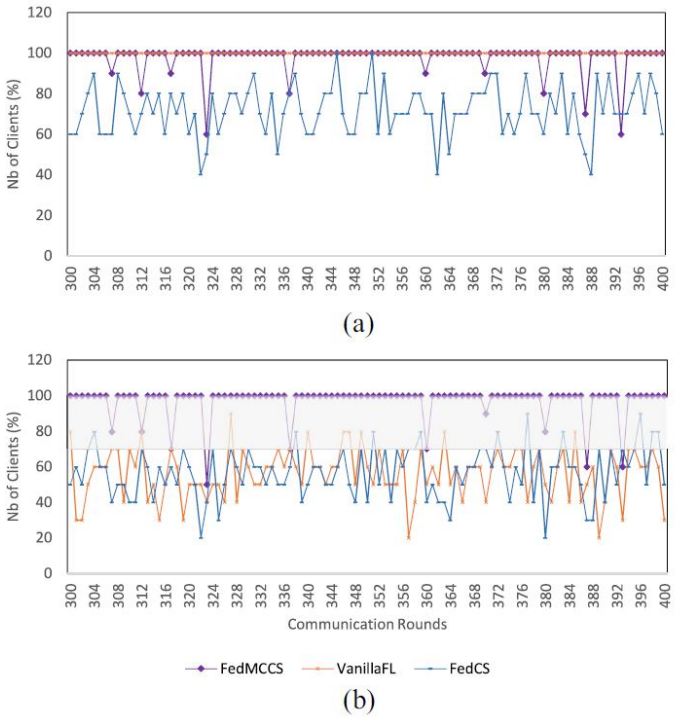


Figure 10: Scheduling (a) and participation (b) rate achieved by FedMCCS in comparison to other baselines [5]

representative of the population data distribution), with its' samples pooled from randomly selected devices. The virtual client is then used to assign the non i.i.d degrees to the different clients in the system. The weight divergence between client k and the virtual client C_0 is given by:

$$d_k(t) = \frac{\|\mathbf{w}_k(t) - \mathbf{w}_0(t)\|}{\|\mathbf{w}_0(t)\|}, \tag{15}$$

where $\mathbf{w}_0(t)$ is the model of the virtual client. A followed rule of thumb is that a large value of weight divergence of device k is induced by a high degree of non-i.i.d of its dataset. The chosen clients are then selected in an increasing order according to their weight divergences.

3.4 Client Selection and Bandwidth Allocation in Wireless Federated Learning Networks

In [19], the clients selection and the bandwidth allocation is jointly optimized. The authors focus on which clients are selected and how much bandwidth is allocated to them, given the devices energy limitations. To this end, the authors propose some experiments to understand the effect of adopting three different client selection schemes. Mainly, the ascending, descending and uniform selection patterns. More specifically, the *descending/ascending pattern* schedules more/less clients at the early stages of training, while maintaining a smaller/larger participation rate at later stages. The *Uniform pattern* selects the same number of clients in each round. Fig. 11 illustrates the testing accuracy evolution as a function of the number of rounds and the client selection patterns that could be followed, which appear to affect the model performance. According to the experiments results shown in Fig.11, the best performance

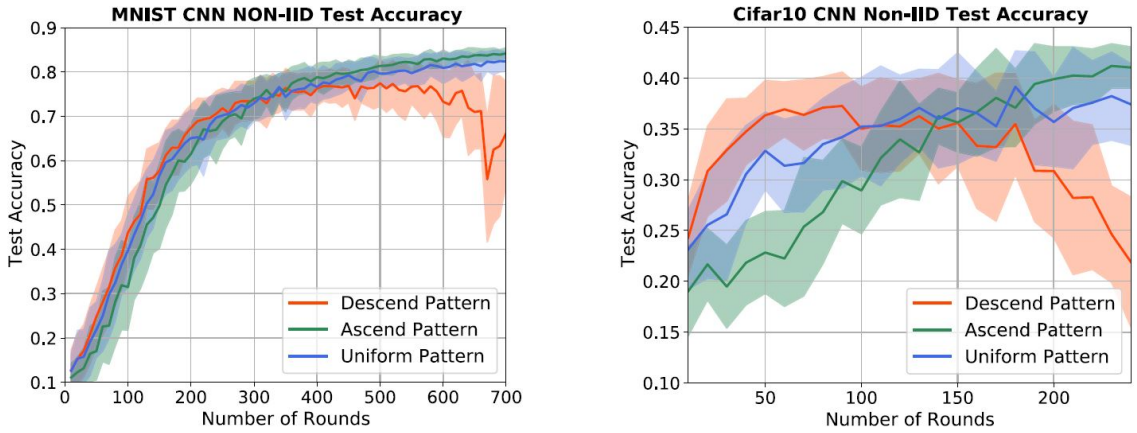


Figure 11: Testing accuracy as a function of the number of rounds and different client selection patterns [19]

shown is attained following the *ascending pattern*. This result stems from the fact that the model during early learning stages can be driven by few number of devices, towards a neighborhood of the global average loss minimizer. The model generalization is later enhanced by scheduling more clients at the end

of the training process to reach the minimizer. To this end, due to the energy limitation of the clients in the system, the authors propose to schedule a small set of devices with large energy budget at the beginning of training, followed by scheduling those with smaller energy budget at later stages. This way, devices with low energy budget are not exhausted early on during training, where their contribution is negligible. In order to realize the aforementioned observations in their problem, the authors propose the following optimization problem:

$$\max_{\mathbf{a}^0,\mathbf{b}^0,\cdots,\mathbf{a}^{T-1},\mathbf{b}^{T-1}} \sum_{t=0}^{T-1} U^t(\mathbf{a}^t) \tag{16}$$

$$s.t. \quad \sum_{t=0}^{T-1} E(a_k^t, b_k^t | h_k^t) \leq H_k, \quad \forall k \tag{17}$$

$$b_{min} \leq b_k^t \leq 1, \quad \forall k, \forall t, \quad \sum_{k=1}^{K} b_k^t = 1 \tag{18}$$

$$a_k^t \in \{0, 1\}, \quad \forall k, \forall t, \tag{19}$$

where

$$U^t(\mathbf{a}^t) = \eta^t \sum_{K=1}^{K} D_k a_k^t$$

is a proxy of the expected FL performance at round t given a set of scheduled users. a_k^t is a scheduling binary random variable of user k at communication round t, D_k is user k dataset size and η^t is a hyper-parameter, which emphasizes the importance of scheduling more devices at round t. Generally, an increasing value of η w.r.t the number of communication rounds would yield better FL performance, as more importance is given to scheduling users at later stages of training. The objective function in (16) aims at maximizing the FL performance throughout the training process. In (17) H_k denotes the energy budget constraint of each client k. E is the energy consumption for each communication round for each user k given his channel gain h_k. If more clients are selected at the early stages of training, the bandwidth allocated for each clients will decrease, hence each client will need more power to send the model updates

to the server to compensate to the lower rates and subsequently, more energy will spent. To address this problem the authors proposed to create a virtual deficit queue $q_k(t)$, which shows the difference of energy spent during training and energy budget of each client. The virtual queue is set to zero at the start of the training process for all clients k, i.e. $q_k(0) = 0, \forall k$. Since it is impossible to know the future CSI in advance, the problem in (16) is reformulated as a per round problem, where T rounds are divided into N frames with R rounds, such that $T = NR$. Within each frame, the CSI are considered to be known and constant. The reformulated problem is then given according to:

$$\min_{\mathbf{a}^t, \mathbf{b}^t} V \cdot U^t(\mathbf{a}^t) - \sum_{k=1} K q_k(t) E(a_k^t, b_k^t | h_k^t) \tag{20}$$

$$s.t. (18), (19), \tag{21}$$

where V is the parameter that controls the trade off between the energy consumption and the number of selected clients. To solve this surrogate problem, *set expansion algorithm (SEA)* is used, where the clients are selected according to their selection priority $\rho_k = \frac{q_k(t)}{(h_k^t)^2}$. The priority is a function of the virtual queue size of each client and their channel quality. The queue length indicates the deviation of the current energy consumption of client k from its long-term energy constraint H_k. The clients are added one by one in an increasing order of ρ_k. A smaller value of ρ_k indicates high priority to select client k. Therefore, it is of interest to schedule those devices that can meet their long term energy constraint in contrast to those that already deviated from it. Moreover, the selection policy emphasizes the importance of scheduling clients with good channel gains in order to conserve their transmission energy.

Based on the work in [19], in [10], a clients selection and bandwidth allocation scheme is proposed to minimize the latency of each round, where the cost function is defined as follows:

$$F(\mathbf{a}^t, \mathbf{b}^t) = T(\mathbf{a}^t, \mathbf{b}^t) - \Phi(\mathbf{a}^t). \tag{22}$$

T is the total latency in each round t given a binary scheduling vector $\mathbf{a}^t$, and a resource allocation vector $\mathbf{b}^t$. $\Phi(\mathbf{a}^t)$ denotes the FL accuracy, which depends

on the scheduling variables and the clients dataset sizes in the following way:

$$\Phi(\mathbf{a}^t) = \sum_{k=1}^{K} \log(1 + \mu D_k(t) a_k(t)), \tag{23}$$

where μ and $D_k(t)$ are the system parameter and dataset size of each client respectively. The optimization problem is given according to:

$$\min_{a_0, b_0, \cdots, a_{T-1}, b_{T-1}} \frac{1}{T} \sum_{t=0}^{T-1} F(a^t, b^t), \tag{24}$$

$$s.t. \quad (17), (18), (19) \tag{25}$$

Similar to [19], the goal is to minimize the long-term cost function over T communication rounds and likewise, the optimization problem is then reformulated as per round problem, dividing the T rounds into N frames, and using the energy deficit queue algorithm called *Per-round Energy Drift Plus Cost algorithm (PEDPC)*, where the objective function is minimized via alternating optimization: First b^t is fixed and a^t is optimized. After optimization, the optimized value of a^t is fixed and used in the optimization of b^t.

In Fig. 12, a comparison among the performance of five different scheduling and resource allocation algorithms, in i.i.d and non-i.i.d settings is illustrated. *PEDPC* is the proposed algorithm. *Select all* selects all clients an equally allocates the bandwidth among them. For the other algorithms the average of 40 clients are selected for training process. *Greedily* selects the clients which satisfy the long-term energy limit requirement according to:

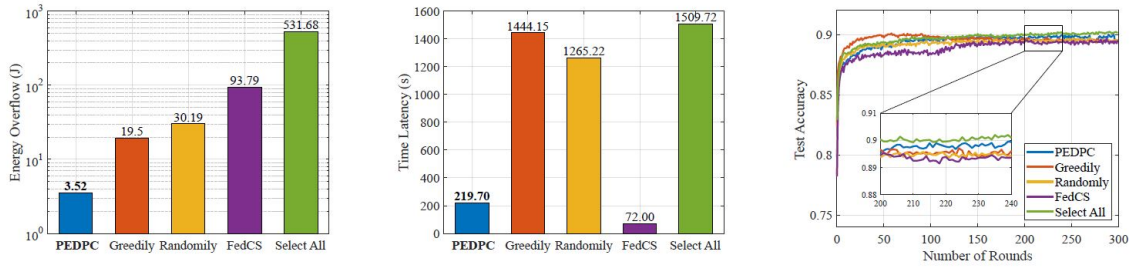
$$\max_{\mathbf{a}^t, \mathbf{b}^t} \sum_{k=1}^{K} a_k^t \tag{26}$$

$$s.t. \quad a_k^t E_k(t) \leq \frac{H_k}{T}, \quad \forall k, \forall t \tag{27}$$

$$(17), (18), (19) \tag{28}$$

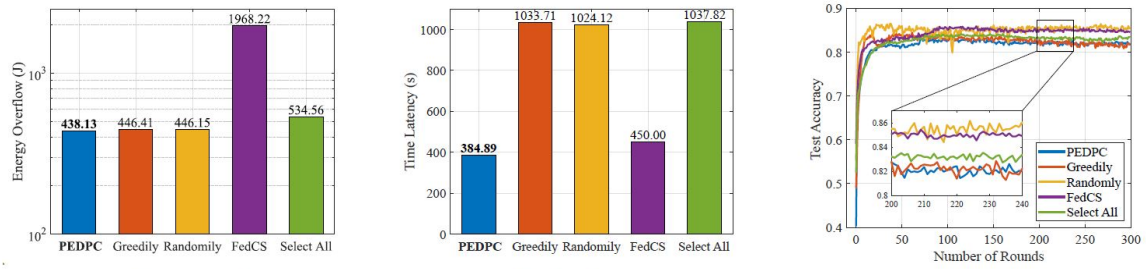
Randomly selects clients in a random way with a given probability p, and allocates bandwidth equally among them. Energy overflow is the difference

between energy consumption and budget. It is obvious that the proposed algorithm in i.i.d case saves more energy compared to the other algorithms. FedCS saves more latency, however it it consumes more energy than PEDPC, because FedCS doesn't take into account the energy consumption. In Fig. 13 the non i.i.d case is shown, where the average number of clients is 90 for all algorithms, except Select all.



(a) Energy overflow (b) Time Latency (c) Test accuracy

Figure 12: Performance metrics evaluation among different approaches in i.i.d setting [10]



(a) Energy overflow (b) Time Latency (c) Test accuracy

Figure 13: Performance metrics evaluation among different approaches in non i.i.d setting [10]

The accuracy in non i.i.d case does not differ a lot. PEDPC still has the advantage in terms of latency, but it consumes almost the same amount of energy compared to the other algorithms. With increasing number of clients FedCS does not have the advantage of low latency.

4. Heterogeneity in Federated Learning

As previously discussed, the deployment of federated learning in wireless networks is significantly hindered by the challenges posed by data and system heterogeneity. For instance, conducting more local epochs during each communication round can reduce the communication burden of frequent model updates, however it may also increase the risk of training failure due to the deviation of the global objective from its minimum. Within this section, we will explore a variety of frameworks that address the challenges posed by data heterogeneity in federated learning.

4.1 FedProx

In [13], a new framework was proposed called FedProx, which enables variable amount of work to be done at each device in terms of local training at each communication round with the goal of tackling the variable computing resources available at the devices. Unlike FedAvg, FedProx allows the aggregation of partial solutions from the stragglers instead of simply drop them out from the aggregation step. This is achieved by introducing a proximal term into the local objective function as follows:

$$h_k(w; w_t) = \underbrace{F_k(w)}_{\text{Traditional Loss}} + \underbrace{\frac{\mu}{2}\|w - w_t\|^2}_{\text{Proximal term}} \tag{29}$$

where w^* is an $\gamma_k^t \in [0, 1]$ inexact solution, which minimizes $\min_w h_k(w; w_t)$ given that

$$\|\nabla h_k(w^*; w_t)\| \leq \gamma_k^t \|\nabla h_k(w_t; w_t)\| \tag{30}$$

and

$$\nabla h_k(w; w_t) = \nabla F_k(w) + \mu(w - w_t) \tag{31}$$

The addition of the proximal term to the local loss and the use of a stopping criteria defined by (30) provide two main benefits. Firstly, it helps to tackle the issue of statistical heterogeneity by constraining the distance between the update from each learner and that of the population. Additionally, it enables the

integration of varying amounts of local training at the devices by adjusting γ_k for each device, thereby reducing the impact of system resource heterogeneity.

4.2 Model contrastive learning (MOON)

While FedProx offers some advantages on non-i.i.d data, it does not perform well on image datasets. To overcome this limitation, a new framework called MOON is proposed in [12], which is based on the model contrastive learning approach. Contrastive learning is primarily used in unsupervised learning for learning visual representations and emphasizes that representations derived from different images should be distinct, while representations derived from the same image should be related. Similar to FedProx, MOON is based on FedAvg. The proposed framework aims to minimize the gap between the representation learned by the local model and the representation learned by the global model, while maximizing the difference between the representation learned by the local model and the representation obtained by the previous local model.

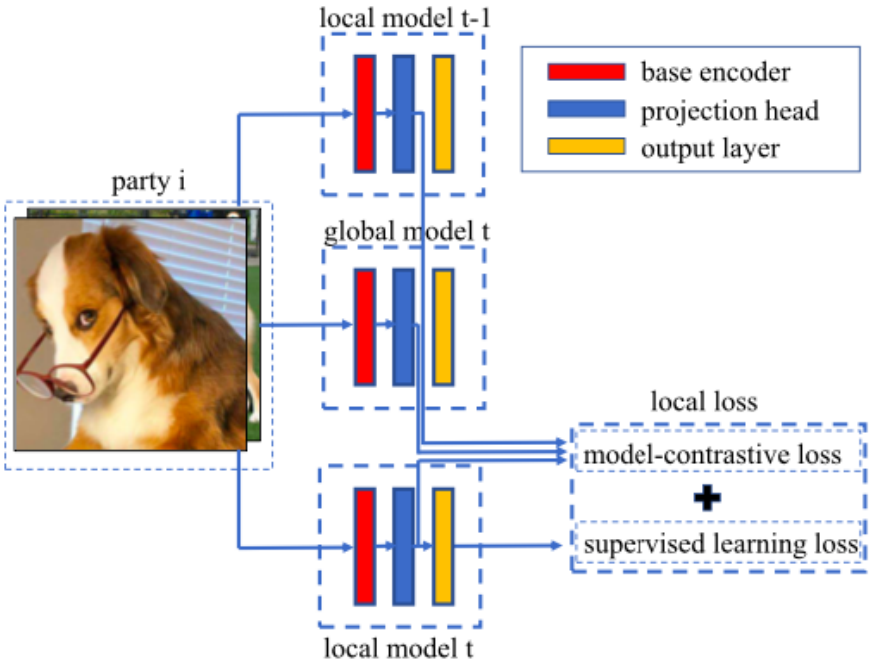


Figure 14: MOON network architecture [12]

The architecture of MOON consists of following three components:

1. *Base encoder:* serves the purpose of extracting representation vectors from input data.
2. *Projection head:* converts the representation into a space of fixed dimensions.
3. *Output layer:* generates the predicted values for each classification.

The local loss which needs to be minimized is given in Eq. (32).

$$\ell_{total} = \underbrace{\ell_{sup}(w_i^t; (x, y))}_{\text{supervised learning loss}} + \underbrace{\mu \ell_{con}(w_i^t; w_i^{t-1}; w^t; x)}_{\text{contrastive loss}} \tag{32}$$

where ℓ_{sup} is the general local loss, and ℓ_{con} is the model contrastive loss given by:

$$\ell_{con} = -\log\left(\frac{\exp(sim(z, z_{glob})/\tau)}{\exp(sim(z, z_{glob})/\tau) + \exp(sim(z, z_{prev})/\tau)}\right), \tag{33}$$

where τ is the temperature parameter, $sim(\cdot, \cdot)$ is a similarity function (e.g. cosine similarity), $z_{glob} = R_{w^t}(x)$, $z_{prev} = R_{w_i^{t-1}}(x)$ and $z = R_{w_i^t}(x)$. $R_w(w)$ is the mapped representation of the input x retrieved at the second-last layer of the network. As the global model is expected to produce more optimal representations, the objective is to minimize the distance between z and z_{glob} while simultaneously increasing the distance between the current and the previous local model representations defined by z and z_{prev} respectively.

The top-1% validation accuracy for the different algorithms as a function of the number of communication rounds is illustrated in Fig. 15. FedProx performs similarly to FedAvg, which highlights its shortcoming in image recognition tasks. Moreover, it's evident that MOON and FedAvg exhibit similar convergence rates. However, MOON is able to attain a higher level of accuracy thanks to added regularization term, characterized by the model contrastive loss which is added to the local loss function, which additionally encourages similar representation learning from augmented data. As a result MOON outperforms other learning methods on a range of image classification tasks, however, it could also be applied to non-image related classification tasks, where

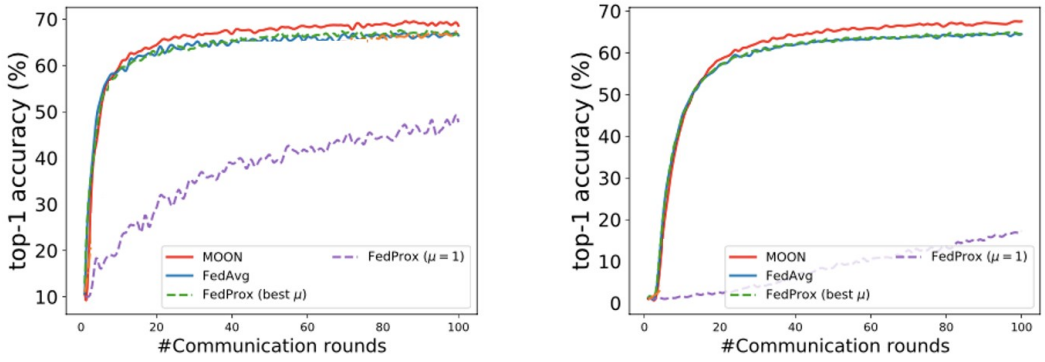


Figure 15: Top-1 accuracy among the different approaches during the training of CIFAR-10 and CIFAR-100 tasks [12]

structural similarities among the input data can be characterized and therefore exploited via contrastive learning.

4.3 Clustering in Federated Learning

As mentioned earlier, the existence of non-i.i.d. data represents a challenge in developing a joint model under FL training. The data heterogeneity in FL can be characterized in the following ways : [6]:

- Input features are not uniformly spread out among the clients.
- Data labels are not uniformly distributed among clients.
- Same input features, but different labeling, or same labeling, but different input features. This phenomena is also known as *Concept Shift*.

In the case of highly non-i.i.d data, training of one global model is not feasible, as it will not be able to fit the objectives of all participating clients [16]. Rather than relying on a single model it may be more advantageous to train several models for different groups of clients with comparable data distributions. To achieve this, a Hierarchical Clustering (HC) approach was proposed in [6]. In particular, HC aims at clustering clients based on their objectives

similarity. Clustering is achieved during a specific communication round, where the aggregated model updates are utilized to determine the similarity between participating clients. Once the clusters are formed, separate models are trained independently but concurrently within the different clusters. During training, the most similar clusters are merged iteratively, combining clients who share similar characteristics and updates. The merging of clusters continues until no notable similarity among clusters is found.

The performance of the proposed algorithm is mainly dependent on how well those clusters are formed. However, if clusters are poorly defined, the algorithm's performance may affect the accuracy.

The experiments are carried out to assess the efficiency of the proposed approach, where FedAvg is used as a baseline. In the an i.i.d setting, as expected, the HC approach was not able categorize clients into clusters, therefore producing a single joint model. This stems from the similarity of the clients generated updates. To asses the performance of HC in non-i.i.d. settings, experiments are considered under pathological (each client randomly receives samples belonging to two labels) and label swapped (each client has two digits labels exchanged with one another) non-i.i.d setting. The evaluation of the test accuracy and the number of clients attaining their target accuracy over each communication round in comparison to FL for the non-i.i.d settings are illustrated in Fig. 16 and 17.

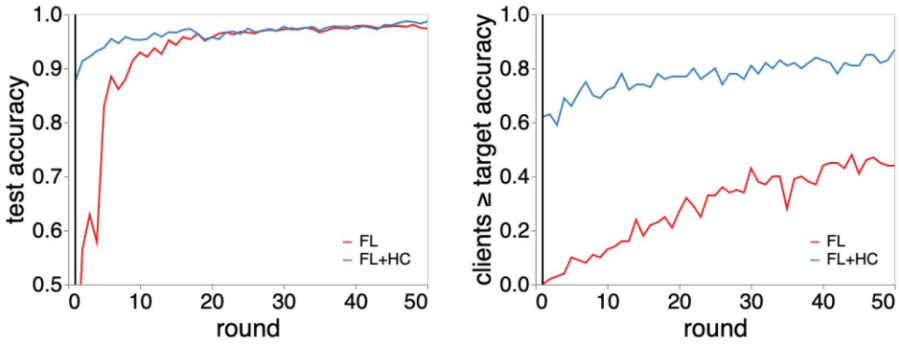


Figure 16: Pathological non-i.i.d setting, clustering step at round=1 [6]

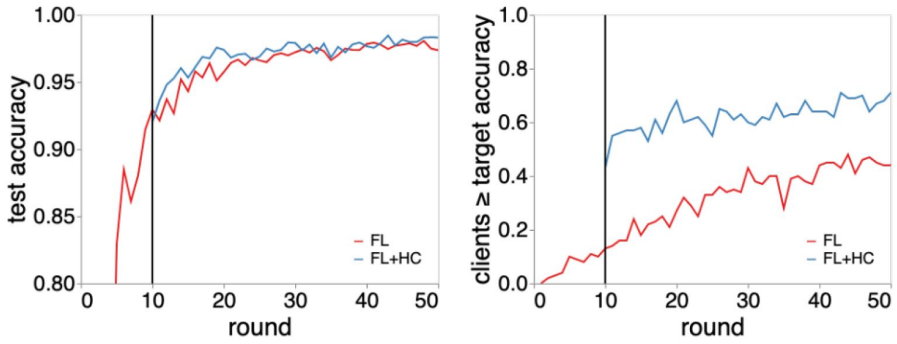


Figure 17: Pathological non-i.i.d setting, clustering step at round=10 [6]

The black horizontal line indicates the communication round where clustering is done. FL represents the FedAvg baseline, and FL+HC is the proposed approach. The largest enhancement in contrast to FL, seems to occur when the clustering step is made during the first communication round. In the label

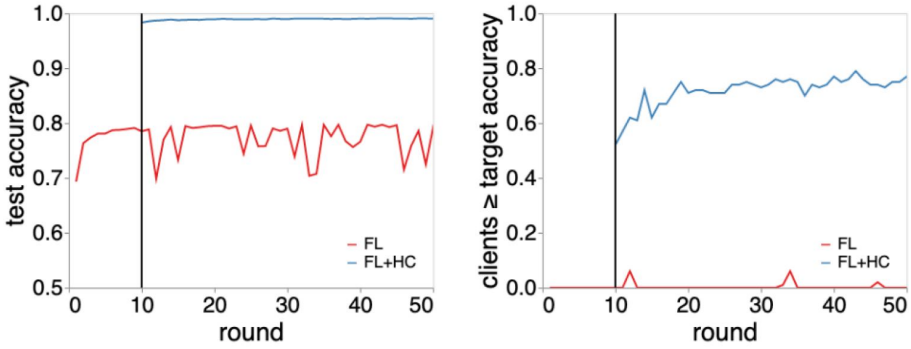


Figure 18: Label swapped non-i.i.d setting, clustering step at round=10 [6]

swapped non-i.i.d setting shown in Fig. 18, FL+HC is able to attain a maximum test accuracy of 80%.

The performance demonstrates that FL with HC converges faster, and results in greater number of clients achieving the desired accuracy in each communication round during severe non-i.i.d settings.

5. Communication-efficient Federated Learning

As mentioned previously, FL faces a significant obstacle in terms of communication and computation costs. Generally, a moderately large number of local epochs aids in decreasing the communication cost. However, this results in an increase of the computational cost, and risks the divergence of the global model in non-i.i.d settings. Fig. 19 illustrates a general trend, in terms of trade-off between the cost of local computation and communication in FedAvg, as a function of the number of local epochs. To enhance the training efficiency it is

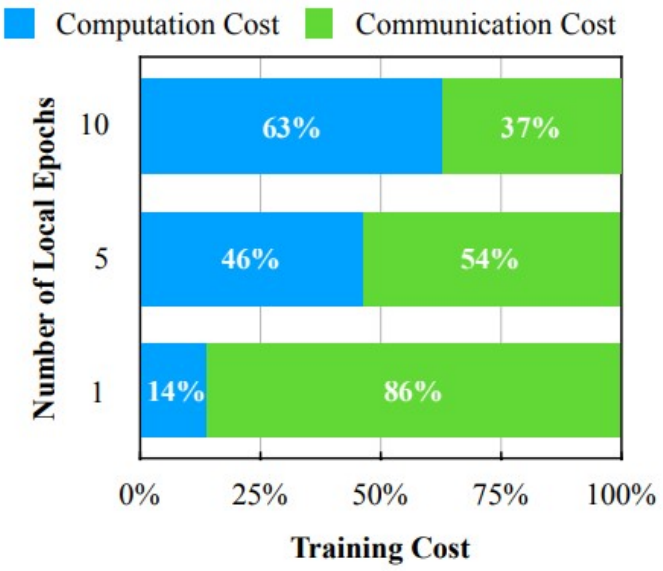


Figure 19: Training cost under varying number of local epochs [11]

necessary to minimize both communication and computation costs. Previous works [22] highlight the existance of subnetworks within the initialized models that are able to generalize well, if found, without any weight modification. Such sub-networks can attain high accuracy and guarantee reduced communication, computation and memory cost.

5.1 FedMask

A new approach is proposed in [11] called FedMask. In FedMask, each device attempts to find a personalized sub-network which generalizes to its' local task, within the initialized global model shared by the server. The personalized sub-networks are found by training a deterministic binary mask (in contrast to training the model weights in classical FL approaches) based on the clients local datasets. The trained masks are then sent back to the server, where similar masks are aggregated together to preserve personalization. Then, the aggregated personalized masks are sent back to their corresponding clients for further training.

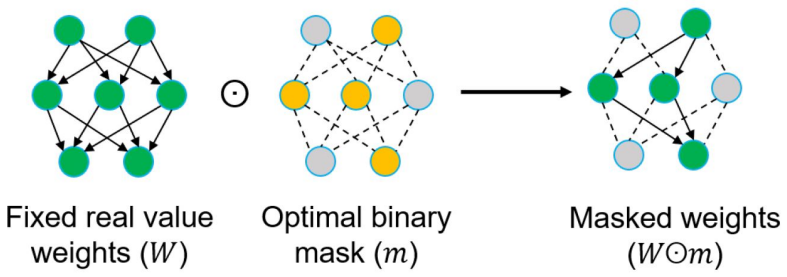


Figure 20: Weight masking process [11]

The training process for FedMask involves the following steps:
Initialization: The server distributes the initialized model and deterministic mask to the clients, and each client keeps the received model frozen throughout the training process.
One shot pruning: Each participating device then learns a heterogeneous binary mask using a one-shot pruning method, which is based on a combination of real-valued masks and fixed weights. This pruning technique helps to remove less important model parameters while preserving the most important ones. During weight pruning, each device will keep the structure of the upper layers and only perform pruning on the classifier layers.
Training and Aggregation: Local training is carried out on the heterogeneous binary masks using the local datasets for several epochs. The result-

ing masks, which embody the personalized sub-networks, are then sent to the server, where similar masks are aggregated together to preserve personalization. The personalized models are then sent back to their corresponding clients. Those steps iterate until the final personalized sub-networks are found.
In Fig. 20 the weight masking process is illustrated, where a sub-network is formed by applying a binary mask $\mathbf{m} \in \{0,1\}^{m \times n}$ to the matrix representing the model connections with fixed weights, using element wise multiplication, such that the output of the produced sub-network can be given by $\mathbf{y} \in \mathbb{R}^m$ defined as follows:

$$\boldsymbol{y} = (\boldsymbol{W} \odot \boldsymbol{m}) \cdot \boldsymbol{x}, \tag{34}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the input, and $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ denotes the weight matrix of the fully connected initialized network. Due to the limitations of current optimization algorithms like SGD, it is not practical to use them on binary masks. As a solution, a new real-valued mask $\boldsymbol{m^r} \in \mathbb{R}^{m \times n}$ has been proposed. Binarization of $\boldsymbol{m^r}$ to $\boldsymbol{m}$ during feedforward step is performed using a sigmoid function and a threshold function as shown in (36)

$$m_{ij} = \sigma(m^r_{i,j}) \in [0,1] \tag{35}$$

and then binarizing $m_{i,j}$ using a threshold operation according to:

$$m_{ij} = \begin{cases} 1, & m_{ij} \geq \tau \\ 0, & m_{ij} \leq \tau \end{cases} \tag{36}$$

where m_{ij} is the element located in the i-th row and j-th column. (37) is utilized to compute the gradients of the loss with respect to the mask m during the back-propagation process.

$$\frac{\partial L}{\partial \boldsymbol{m}} = \left(\frac{\partial L}{\partial \boldsymbol{y}} \cdot \boldsymbol{x}^T \right) \odot \boldsymbol{W}, \tag{37}$$

However, due to the fact that threshold function is not differentiable, the gradient of the loss w.r.t $\boldsymbol{m}^r$ is computed and then the sigmoid function $\sigma(\cdot)$ is

used to obtain $m_{ij} = \sigma(m_{ij}^r)$. The gradient w.r.t $\boldsymbol{m}^r$ could be computed using the chain rule, and is given by:

$$\frac{\partial L}{\partial \boldsymbol{m}^r} = \boldsymbol{\Phi} \odot \frac{\partial L}{\partial \boldsymbol{m}}, \tag{38}$$

where $\boldsymbol{m} = \sigma(\boldsymbol{m}^r)$ and $\boldsymbol{\Phi} = \boldsymbol{m} \odot (1 - \boldsymbol{m})$ denotes the derivative of the sigmoid function.

In FedMask, the aggregation process cannot be performed through simple averaging, due to heterogeneity of the binary masks with non-overlapping elements. Given the personalization goal of FedMask, the proposed approach should aim at preserving the personalized structure and information during the aggregation step. Accordingly, the authors propose a novel aggregation policy to aggregate the personalized binary masks. Averaging is made element-wise. Model elements could be composed of channels, full layers, or single weights. The authors propose to average the elements present in at least two binary masks. The central server then updates the mask with the aggregated values and keeps the elements that are unique to a single mask unchanged. As shown in Fig. 21, the

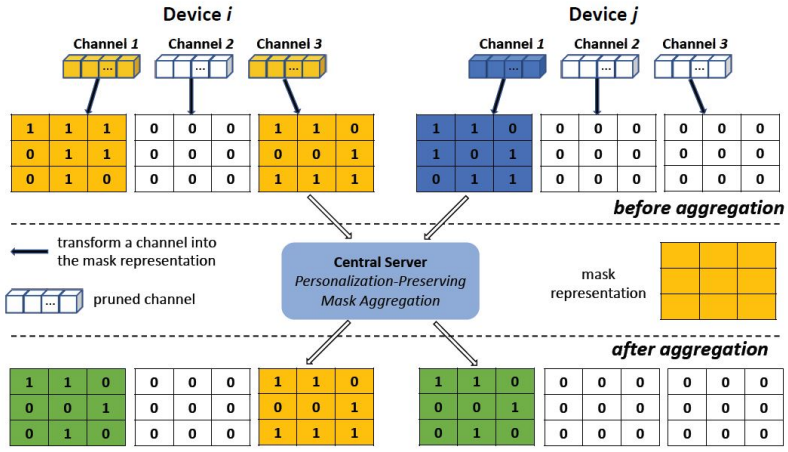


Figure 21: Personalization-preserving mask aggregation. The unpruned masks are depicted using yellow and blue matrices, while the updated masks resulting from the intersection between devices are shown in green. [11]

first channel of device i and device j are intersected (shown in yellow), which

causes the proposed aggregation technique to average the masks of these two channels (shown in green). The white channels however represent the pruned ones, consequently they will remain unchanged for both users. Consequently, two separate personalized binary masks are then produced via thresholding and transmitted to their corresponding devices.

5.2 FedPM

A similar approach is proposed in [9] called Federated Probabilistic Mask Training (FedPM). The proposed approach is illustrated in Fig. 22. The server

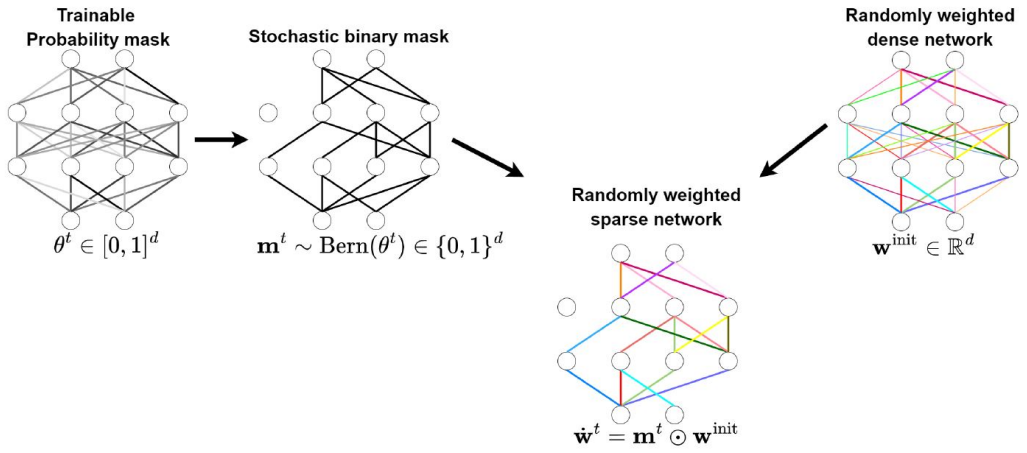


Figure 22: Generation of a sparse sub-network with random weights, achieved through a trainable probability mask. [9]

initializes the model randomly $\boldsymbol{w}^{\boldsymbol{init}}$, and transmits the initialized model in addition to a probability mask $\boldsymbol{\theta} \in [0,1]^d$ to all clients. Each client then generates a score mask $s \in \mathbb{R}^d$ using the inverse sigmoid function according to:

$$\boldsymbol{s} = \sigma^{-1}(\boldsymbol{\theta}), \tag{39}$$

Then, a binary mask $\boldsymbol{m}$ is generated using the Bernoulli distribution over the transmitted probability mask $\boldsymbol{\theta}$, $\boldsymbol{m} \sim Bern(\boldsymbol{\theta})$. Then, each client performs model sparsification $\boldsymbol{w} = \boldsymbol{m} \odot \boldsymbol{w}^{init}$ to produce it's own stochastic sub-network,

which is used for the feed-forward computations over the local dataset. Given that the sampling operation $\boldsymbol{m} \sim Bern(\boldsymbol{\theta})$ is not differentiable, error back-propagation is performed to update the score masks instead of updating the binary masks directly. This is done locally at each client via gradient descent methods according to:

$$\boldsymbol{s} = \boldsymbol{s} - \eta \nabla \mathcal{L}(f_{\boldsymbol{w}}, \mathcal{D}_k), \tag{40}$$

The clients then extract the locally updated probability masks by applying the sigmoid function according to :

$$\boldsymbol{\theta} = \sigma(\boldsymbol{s}) \tag{41}$$

Then, a new binary mask is sampled from the newly produced probability mask at each client. The sampled binary masks are sent in the UL to the server, which in turn averages to estimate the global probability mask given by: $\bar{\theta}^{g.t}$ as:

$$\hat{\bar{\boldsymbol{\theta}}} = \frac{1}{K} \sum_{k \in \mathcal{K}_t} \boldsymbol{m}_k, \tag{42}$$

which is unbiased estimation of true mean. The global probability mask is then sent in the DL to all users. Those steps iterate until a global probability mask is found, which is able to produce stochastic sub-networks fit to the clients datasets. In the experimental evaluation, FedPm attains higher validation accuracy in comparison with FedMask and other baselines, despite the fact that FedMask converges faster as shown in Fig. 23. Additionally,as illustrated in Fig. 24 FedPm proves to be more communication efficient in comparison to other. FedMask requires 1 bit per parameter (bpp) when communicating binary mask to the server, meanwhile in FedPm the communication can be reduced lower that 1bpp, due to random masking process and the unequal distribution of ones and zeroes within the binary masks.

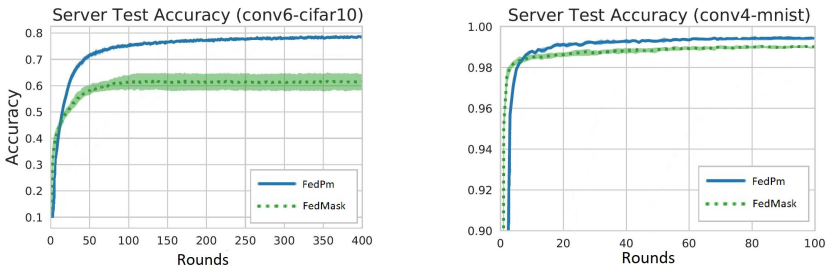


Figure 23: Accuracy comparison between FedPm and FedMask

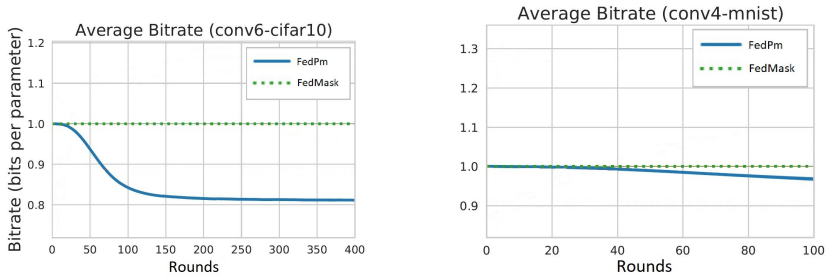


Figure 24: Bitrate comparison between FedPm and FedMask

6. Conclusion

Based on the discussions presented in this survey, it is evident that FL can be considered as an efficient solution to the challenges faced by traditional cloud-based ML approaches, such as poor communication efficiency, high latency, and privacy concerns. The FL approach allows devices to use their local data to train a ML model in a decentralized way, iteratively optimizing the model locally and sharing the resulting model updates to a centralized server for aggregation.

One of the key challenges in implementing FL on a large scale is the communication bottleneck in wireless resources. This survey discussed the compression techniques, such as sparsification and quantization, which aid in countering the unfavorable communication burden in FL settings. Additionally, AirComp technique was explored, which can further improve the efficiency of FL in wire-

less networks compared to the widely used digital OFDMA, by leveraging the superposition feature of radio waves to combine and average uncoded analog modulated model updates transmitted by users. Furthermore, an overview of different client selection frameworks for FL is provided. Moreover, the detrimental effect of data heterogeneity on FL training was discussed, as well as the proposed FL frameworks which aid in deterring its effects, such as FedProx and MOON. Additionally, the potential of clustering approach in Federated Learning in addressing severely heterogeneous datasets was highlighted.

Finally, this work covered two novel approaches, FedMask and FedPm, which enable collaboration among clients to find underlying sub-networks within the initially initialized model, that are able to generalize well over a given task. The discovery of those sub-networks involve binary mask training in lieu of the traditional model training. As a result, in this setting, clients enjoy communication efficient collaboration during training.

In conclusion, this survey paper provides a comprehensive overview of the fundamentals and implementation challenges of Federated Learning, covering various techniques and frameworks that address communication efficiency, client selection, and data heterogeneity.

References

[1] Sawsan Abdulrahman, Hanine Tout, Azzam Mourad, and Chamseddine Talhi. Fedmccs: Multicriteria client selection model for optimal iot federated learning. *IEEE Internet of Things Journal*, 8(6):4723–4735, 2021. doi: 10.1109/JIOT.2020.3028742.

[2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *CoRR*, abs/1704.05021, 2017. URL `http://arxiv.org/abs/1704.05021`.

[3] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Sarit Khirirat, Nikola Konstantinov, and Cédric Renggli. The convergence of sparsified gradient methods. *CoRR*, abs/1809.10505, 2018. URL `http://arxiv.org/abs/1809.10505`.

[4] Mohammad Mohammadi Amiri, Deniz Gündüz, Sanjeev R. Kulkarni, and H. Vincent Poor. Convergence of federated learning over a noisy downlink. *IEEE Transactions on Wireless Communications*, 21(3):1422–1437, 2022. doi: 10.1109/TWC.2021.3103874.

[5] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019. URL `http://arxiv.org/abs/1902.01046`.

[6] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. *CoRR*, abs/2004.11791, 2020. URL `https://arxiv.org/abs/2004.11791`.

[7] Mingzhe Chen, Zhaohui Yang, Walid Saad, Changchuan Yin, H. Vincent Poor, and Shuguang Cui. A joint learning and communications frame-

work for federated learning over wireless networks. *CoRR*, abs/1909.07972, 2019. URL http://arxiv.org/abs/1909.07972.

[8] Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H. Vincent Poor. Distributed learning in wireless networks: Recent progress and future challenges. *CoRR*, abs/2104.02151, 2021. URL https://arxiv.org/abs/2104.02151.

[9] Berivan Isik, Francesco Pase, Deniz Gunduz, Tsachy Weissman, and Michele Zorzi. Sparse random networks for communication-efficient federated learning, 2023.

[10] Yun Ji, Zhoubin Kou, Xiaoxiong Zhong, Sheng Zhang, Hangfan Li, and Fan Yang. Client selection and bandwidth allocation for federated learning: An online optimization perspective, 2022. URL https://arxiv.org/abs/2205.04709.

[11] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys '21, page 42–55, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450390972. doi: 10.1145/3485730.3485929. URL https://doi.org/10.1145/3485730.3485929.

[12] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. *CoRR*, abs/2103.16257, 2021. URL https://arxiv.org/abs/2103.16257.

[13] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[14] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. URL `http://arxiv.org/abs/1602.05629`.

[15] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. *CoRR*, abs/1804.08333, 2018. URL `http://arxiv.org/abs/1804.08333`.

[16] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints. *CoRR*, abs/1910.01991, 2019. URL `http://arxiv.org/abs/1910.01991`.

[17] Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. *CoRR*, abs/1809.07599, 2018. URL `http://arxiv.org/abs/1809.07599`.

[18] Hong Xing, Osvaldo Simeone, and Suzhi Bi. Federated learning over wireless device-to-device networks: Algorithms and convergence analysis. *CoRR*, abs/2101.12704, 2021. URL `https://arxiv.org/abs/2101.12704`.

[19] Jie Xu and Heqiang Wang. Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. *IEEE Transactions on Wireless Communications*, 20(2):1188–1200, 2021. doi: 10.1109/TWC.2020.3031503.

[20] Wenyu Zhang, Xiumin Wang, Pan Zhou, Weiwei Wu, and Xinglin Zhang. Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access*, 9:24462–24474, 2021. doi: 10.1109/ACCESS.2021.3056919.

[21] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018. URL `http://arxiv.org/abs/1806.00582`.

[22] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf.

[23] Guangxu Zhu, Yong Wang, and Kaibin Huang. Low-latency broadband analog aggregation for federated edge learning. *CoRR*, abs/1812.11494, 2018. URL http://arxiv.org/abs/1812.11494.

[24] Guangxu Zhu, Jie Xu, and Kaibin Huang. Over-the-air computing for 6g - turning air into a computer. *CoRR*, abs/2009.02181, 2020. URL https://arxiv.org/abs/2009.02181.