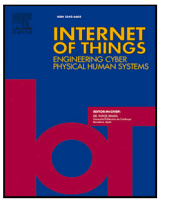
Internet of Things 22 (2023) 100742

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Review article

Communication and computation efficiency in Federated Learning: A survey

Omair Rashed Abdulwareth Almanifi [a], Chee-Onn Chow [a,*], Mau-Luen Tham [b], Joon Huang Chuah [a], Jeevan Kanesan [a]

[a] Department of Electrical Engineering, Faculty of Engineering, Universiti Malaya, Lembah Pantai, 50603 Kuala Lumpur, Malaysia
[b] Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Jalan Sungai Long, Bandar Sungai Long, 43000 Kajang, Selangor, Malaysia

ARTICLE INFO

Keywords:
Federated Learning
Internet of Things
Communication efficiency
Computation efficiency
Machine learning

ABSTRACT

Federated Learning is a much-needed technology in this golden era of big data and Artificial Intelligence, due to its vital role in preserving data privacy, and eliminating the need to transfer and process huge amounts of data, while maintaining the numerous benefits of Machine Learning. As opposed to the typical central training process, Federated Learning involves the collaborative training of statistical models by exchanging learned parameter updates. However, wide adoption of the technology is hindered by the communication and computation overhead forming due to the demanding computational cost of training, and the large-sized parameter updates exchanged. In popular applications such as those involving Internet of Things, the effects of the overhead are exacerbated due to the low computational prowess of edge and fog devices, limited bandwidth, and data capacity of internet connections. Over the years, many research activities that target this particular issue were conducted but a comprehensive review of the fragmented literature is still missing. This paper aims at filling this gap by providing a systematic review of recent work conducted to improve the communication and/or computation efficiency in Federated Learning. We begin by introducing the essentials of Federated Learning and its variations, followed by the literature review placed according to an encompassing, easy-to-follow taxonomy. Lastly, the work sheds light on the current challenges faced by the technology and possible directions for future work.

1. Introduction

It is no secret that technological advancement in computational hardware resulted in a massive boost of interest in Artificial Intelligence (AI) generally and Machine Learning (ML) specifically. Global Corporate investment in 2021 was estimated at about 176.47 billion U.S. Dollars, a growth of about 33.7 times the total investment in the year of 2013 [1]. $93.5 billion of the aforementioned total was privately invested, funding a large number of AI companies including 746 start-ups. These investments helped almost double the number of research work published in the field since 2010, with a total of 334.5 thousand publications seen in 2021 alone, 51.5% of which were peer-reviewed journal papers. This evident growth of capital and research interest accelerated technological development by multiple orders of magnitude, ushering an era where AI has gained a substantial role in all kinds of industries, including healthcare [2], marketing [3], and Internet of Things (IoT) [4,5].

* Corresponding author.
E-mail address: cochow@um.edu.my (C.-O. Chow).

https://doi.org/10.1016/j.iot.2023.100742
Received 29 September 2022; Received in revised form 8 February 2023; Accepted 3 March 2023
Available online 5 March 2023

Nomenclature

Abbreviations

AI	Artificial Intelligence
$AIoT$	Artificial Intelligence of Things
CNN	Convolutional Neural Network
CS	Compressive Sensing
DL	Deep Learning
DQL	Deep Q-learning
DRL	Deep Reinforcement Learning
FL	Federated Learning
$FLOPS$	Floating point operations per second
GPU	Graphics Processing Units
IID	Independent and identically distributed
IoT	Internet of Things
KLD	Kullback–Leibler divergence
MBS	Macro Base Station
ML	Machine Learning
MLP	Multilayer Perceptron
OTA	over-the-air
RNN	Recurrent neural network
SBS	Small Base station
SGD	Stochastic Gradient Descent
$SINR$	Signal-to-Interference-plus-Noise Ratio
SNR	Signal-to-Noise Ratio
TPU	Tensor Processing Units
UAV	Unmanned Aerial Vehicles

The impressive amount of enthusiasm shown in favor of AI is evident of its positive impact on industries and the personal lives of individuals. Nonetheless, this impact comes at a price, which is data. ML is the most commonly used subset of AI where algorithms learn from data examples to make inferences with no guidance or pre-established rules. The most popular and complex type of ML algorithms is Deep Learning (DL), which is characteristically data-hungry [6]. This massive need for data resulted in a surge of threats to the privacy of individuals, spawning a need for legislative regulation to better handle private and sensitive data. Some of these regulations could be seen in China's Personal Information Protection Law of 2021, Europe's General Data Protection Regulation enforced in 2018 [7], South Korea's Personal Information Protection Act of 2011, and the California Consumer Privacy Act introduced in 2018, just to name a few.

Recently, a distributed approach to training ML algorithms was introduced with the motivation of preserving data privacy. Federated Learning (FL) omits the need for sharing data with central ML training servers, by periodically sharing the updated parameters of the ML model [8]. In this configuration, a model is trained locally on the user's (client) side on their own unique, personal dataset, and only the updates to the model are shared with the server. All the communicated updates are aggregated at the server to produce a global model with "experience" from all the clients' datasets. That model is then sent back to the clients for further training or for making inferences. FL is general enough to apply to any ML model, and hence can be used for any type of application with a server–client architecture. Since this is the typical structure of IoT networks and with the popular rise of Artificial Intelligence of Things (AIoT) where AI algorithms are deployed at the edge/fog, FL is seen as a natural evolution for ML in the field of IoT.

Although FL is advantageous when it comes to ensuring the privacy of data, its usage is prone to multiple drawbacks, one of which is the added communication cost of periodically updating the model parameters, as opposed to the one-time transfer of data seen in conventional centralized training [9]. This overhead is exacerbated as the number of clients increases, slowing down the process even further. In many applications of FL, especially those involving IoT, the clients are devices with limited computational capacity such as the micro-controllers and microprocessors used in Unmanned Aerial Vehicles (UAVs), smart home appliances, and smart wearables. While most servers are equipped with state-of-the-art Tensor Processing Units (TPUs) or Graphics Processing Units (GPUs), these client devices are often equipped with far inferior processing units, that are often occupied with other tasks besides model training. The need for solutions to these issues is, therefore, necessary if FL were to replace traditional centralized training.

Plenty of research efforts have been put into finding solutions to the aforementioned challenges since the emergence of FL in 2016, giving birth to a large body of work dedicated to communicational and computational efficiency. This survey looks to provide

a coherent summary of the work conducted on improving communication and computation in the context of FL, as a way to provide interested researchers with an efficient gateway into the field.

1.1. Related surveys review

Even though the advent of FL was very recent, several reviews, surveys, and tutorials were published, giving comprehensive overviews of the advances in the field and the challenges yet to be addressed. Each of these pieces of literature chose to either improve on its predecessors or concentrate on a specific perspective when approaching the field of FL

[10] offered a comprehensive review of various aspects of FL, including the essential architectures and designs categorized into different modeling aspects, the technical considerations taken during deployment, areas of application and utilization, solutions to common issues plaguing FL, and the current challenges and possible future areas of research. The methodology adopted in the survey was systematic and provided a detailed, improved account of the overall landscape of FL as was in the year of 2020. Nonetheless, the paper touched on issues of communication and computation only briefly.

Another survey conducted concentrated concentrate on the practical side of FL, giving a summary of the advances made on the software, hardware, algorithms, security protocols, and platforms utilized for the sake of deploying FL systems in practice [11]. It also introduced examples of real-life applications and the challenges faced by the systems as well as a brief guide on modeling and designing FL architectures. While efficiency was briefly discussed in the paper, it was merely in the context of it being an upside to some FL technologies. Regardless, this work stands as a great introduction to the topic and provides readers with interest in availing themselves of the technology a decent head start.

Zhang et al. produced a tutorial paper on FL, covering it from five different aspects, namely the type of data partitioning, the privacy-preserving techniques, the models and algorithms of machine learning, the communication architectures, and the adopted approaches to tackling the issue of heterogeneity [12]. Similar to the aforementioned papers, this work was concluded after providing a list of application uses for FL, as well as the challenges and open research areas. Despite communication being a central aspect in the paper, the amount of attention given to it was quite small, with little mention of the computational efficiency.

Practical use of FL requires the implementation of practical systems that make use of the theory, a survey that chose to approach FL from the point view of its implemented systems was authored by Li et al. [13]. The work started with an overview of FL systems and their constituent components before delving into a comprehensive taxonomy that covers various aspects of the topic. Furthermore, it introduced multiple examples of popular systems and compared them with sufficient details. The authors also suggested some design considerations followed by case studies of real-life applications in the healthcare, mobile service, and financial industry. Lastly, a spotlight on the future scope of interesting FL research was given.

While the work above provided a literature review of the field of FL generally, [14] chose to shift its focus towards a specific aspect: communication efficiency. The review detailed the core research questions raised when addressing communication efficiency and the recent methods proposed to answer them. The solutions were categorized into local updating, selection of clients, model updates reduction, decentralized/peer-to-peer learning, and compression schemes, giving a quick overview of research work and its employed methods. The conclusion of the work provided a discussion of future work and expectations for communication efficiency in FL.

The tutorial in [15] had its concentration put on the concept of IoT in relation to FL, providing a rigorous review of the work and findings achieved thus far in the use of FL over IoT network. The survey sought to cover a larger area in its review by including multiple significant parameters: methods of resource optimization, incentive mechanisms, local machine learning models, global aggregation approaches, design of end-devices and cloud servers, and security, all while devising a coherent taxonomy for FL in IoT. Furthermore, open research challenges and issues were discussed in detail, alluding to possible solutions awaiting research and analysis, thereby concluding the paper with future predictions of the state of FL in the context of IoT.

The reviews summarized above are not a complete account of all existing surveys conducted on FL, much work was left out due to redundancy or lack of direct correlation to the topic of discussion this paper is centered around. Some of such work is presented in [12,16], and [17] which provided a general review of advancement in overall FL, while [18,19] chose to focus on specific aspects of FL.

For the sake of clarity, Table 1 is prepared to provide a systematic comparison between the review work discussed in this section against our paper. As expected, due to the communication and computational challenges in FL not being the focal point of discussion in most of the surveys, a lack of coverage is observed. Although often marginal, client selection and updates compression were the most touched upon communicational challenges in the works, whereas, no survey discussed updates dropping or over-the-air aggregation, presumably owing to them being very recent techniques. In regards to computation, we observe that out of all four challenges only resource allocation was covered, even though a considerable amount of research effort has been put into the other topics, as will be made clear in future sections. This observed absence of discussion drives the need for this paper.

1.2. Motivation and contribution

In contrast to other surveys and tutorials written on the topic of FL, this work stands out in being the first to provide a comprehensive and systematic review of the research conducted to address the issue of communication and computation efficiency in the context of FL. As seen in Table 1, while some literature touched on efficiency in FL, in most cases it was either too brief or

Table 1
Summary of the communicational and computational challenges reviewed in recent literature.

Ref.		[10]	[11]	[13]	[14]	[15]	[16]	[12]	[17]	[18]	[19]	Ours
Focus		General FL	General FL	Systems of FL	Communication efficiency	FL over IoT networks	General FL	General FL	General FL	FL over IoT networks	FL over IoT networks	Communication & computation efficiency
Communication Challenges	Client Selection	✓	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓
	Updates Compression	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓
	Updates Dropping	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Network Topology	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓
	Over-the-air Aggregation	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Computation Challenges	Resource Allocation	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓
	Data Quality	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Hyper-parameter Optimization	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Model Pruning and distillation	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Remark		Very few papers on communi-cation	Marginal discussion on efficiency	–	Marginal discussion on efficiency	–	–	–	–	–	Only one paper on client selection	

with a lack of comprehensive focus. Although [15] is primarily a reviewed work on communication efficiency, it has omitted the discussion of computation efficiency.

Primarily FL differs from conventional ML in its heavy reliance on communication channels during the process of training, and the utilization of often ill-prepared local devices to perform the highly complex computations of ML algorithms. This is particularly evidenced by the wide interest in its employment for IoT and AIoT applications, where the clients are underequipped for the communicational and computational cost of FL training. More often than not, attempts at improving one of the two aspects of the technology result in disadvantageous side effects on the other. This tradeoff has been noted by much of the literature reviewed in this paper and omitting it from the discussion produces impractical solutions only. Therefore, we sought to cover both aspects of the FL efficiency problem, out of the belief that the nuance that this brings forth is worthy of exploration.

Furthermore, the notion of FL remains very young in prospect and is rapidly evolving with many advancements taking place within very short intervals of time. Our paper provides an updated view of the current landscape of the work done on improving the efficiency of communication and computation in FL, as well as a general guide for those looking to either expand on the reach of FL or find solutions to issues in their implementations. In other words, our contributions are as follows:

- We provide a comprehensive summary of the latest findings in improving the major issues of overhead in communication and resource allocation in computation efficiency regarding FL implementations and systems.
- We present a systematic categorization of various methods utilized in addressing the issue, with a detailed comprehensive overview of the work and its key contributions.
- We identify the open challenges and unanswered research questions in regard to communication and computation efficiency of FL systems, and propose direction for further research.

1.3. Outline of the paper

As given in Fig. 1, our paper consists of seven sections in total. Section 2 is dedicated to giving a comprehensive summary of the methodology of FL, with significant definitions, a breakdown of its components, and necessary taxonomy. Section 3 focuses primarily on communication efficiency, whereas Section 4 covers computation efficiency; both of which provide essential background, and categorize current areas of research and solutions into convenient subsections. Section 5 goes over the current challenges and the possible research solutions before concluding the paper in Section 6.

2. Federated learning: an overview

FL or collaborative learning is a method used in ML to train an algorithm across multiple decentralized edge devices or servers using their own local data. Although the groundwork for FL was laid long before, the formal introduction of the techniques was credited to Google in 2016 [8,20], which helped formulate the central problem and introduced FL as a viable and reliable solution. Ever since, the concept evolved immensely to accommodate a multitude of variations of the initial issue, as well as to be employed in all types of applications and industries [2–5]. In this section, we introduce the concept of FL, its composing elements, variations, and common framework implementations.

2.1. Definition, background and problem formulation

FL is a restructuring of the training process in ML, in which the need for a unified data center is removed to enhance data privacy preservation. The clients/users that would otherwise communicate their data to the center, collaboratively train the ML model and communicate their local updates to the parameters instead. This shifts the role of the central server to become the instructor that ensures the learning process is conducted properly and the aggregator that unifies all the locally trained models into one global model. This approach has several advantages: aside from enhancing privacy; it reduces the enormous load of training normally put on the server, provides room for local personalization of the model, and eliminates the need to transfer a huge amount of data over the network [21].

2.1.1. Essential components to Federated Learning systems

Due to the rapid development that the techniques have witnessed over the years, there exists plenty of diversity among FL architectures and systems. However, several key components are often seen in any typical FL system; these components are:

- **Model**: the central ML model that the federated process seeks to train. It is constructed at the central server and is communicated to all clients at the start of each training round for joint training.
- **Clients/user equipment**: any computational pieces of hardware tasked with the local training of the model, and communicating parameter updates to the server. In the case of AIoT, for example, the client side is represented by the edge and the fog where AI computations normally take place.
- **Central server/manager**: also known as the cloud in IoT system. It is the instructor of the entire process, where the global model is first designed, and the rules of the training process are defined. Its responsibilities involve selecting clients for participation, receiving parameter updates, and aggregating those parameters to improve the performance of the model. Note that in the case of decentralized FL the server is replaced with a consensus mechanism and is eliminated from the structure.

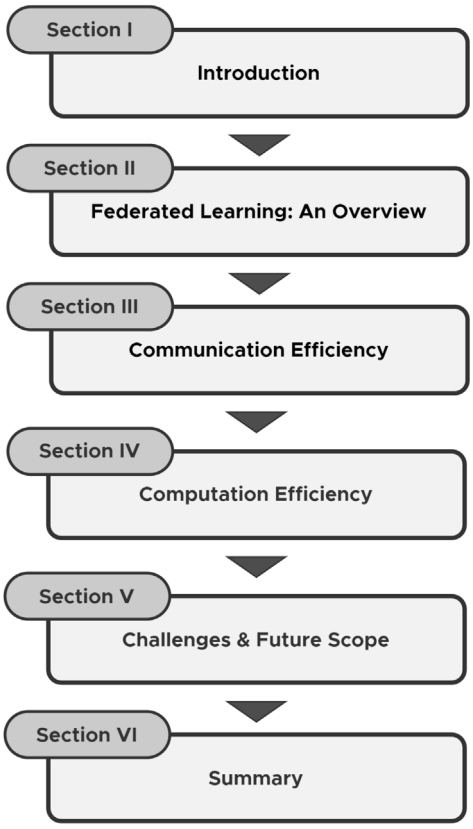


Fig. 1. Overall outline of the paper.

- **Communication channel**: any medium of communication used to communicate instructions and parameter updates between the clients and the server.
- **Parameter updates**: the updates to the parameters of the ML model, learned in the latest round of local training. They are communicated at the end of each training round to the central server where aggregation takes place.

2.1.2. Communication rounds in Federated Learning

A communication or global training round refers to a complete training cycle in which the global model is updated once. Typically, training rounds in FL repeat iteratively until the global model converges and a desired accuracy is achieved. An average FL training cycle can be seen in Fig. 2, and it consists of four stages:

1. **Global Model communication**: Available clients notify the manager of their readiness. The server then communicates the global model to all participating clients signaling the beginning of the communication round.
2. **Local Training**: Each client trains the model locally making use of its own local dataset. The hyperparameters are decided upon either locally or globally depending on the training scheme set up by the server.
3. **Client Selection**: The central manager selects a subset of clients to transmit their updates of the model. The selection protocol is carried out by the server and its significance lies in preventing diminishing returns seen when the full set of clients is used. It should be noted that in some FL frameworks, this step could take place before local training by monitoring the data quality to decrease computational wastage. Works that implemented such a scheme are discussed in 4.3.
4. **Global Aggregation**: Upon reception of the updated parameters from each selected client, the server aggregates them into one global model via a certain aggregation mechanism. With the model updated, the communication round comes to an end. The performance of the model could be assessed and training could be terminated or a new communication round could begin accordingly.

2.1.3. Problem formulation in Federated Learning

In the field of ML broadly, the problem addressed is that of learning a model that maps a set of input statistical data x to a set of desired outputs y based on numerous training examples. The performance of the model is evaluated based on a certain loss function $f_i(w)$, which is an error metric that represents the closeness of the outputs of the model to the actual desired outputs. In

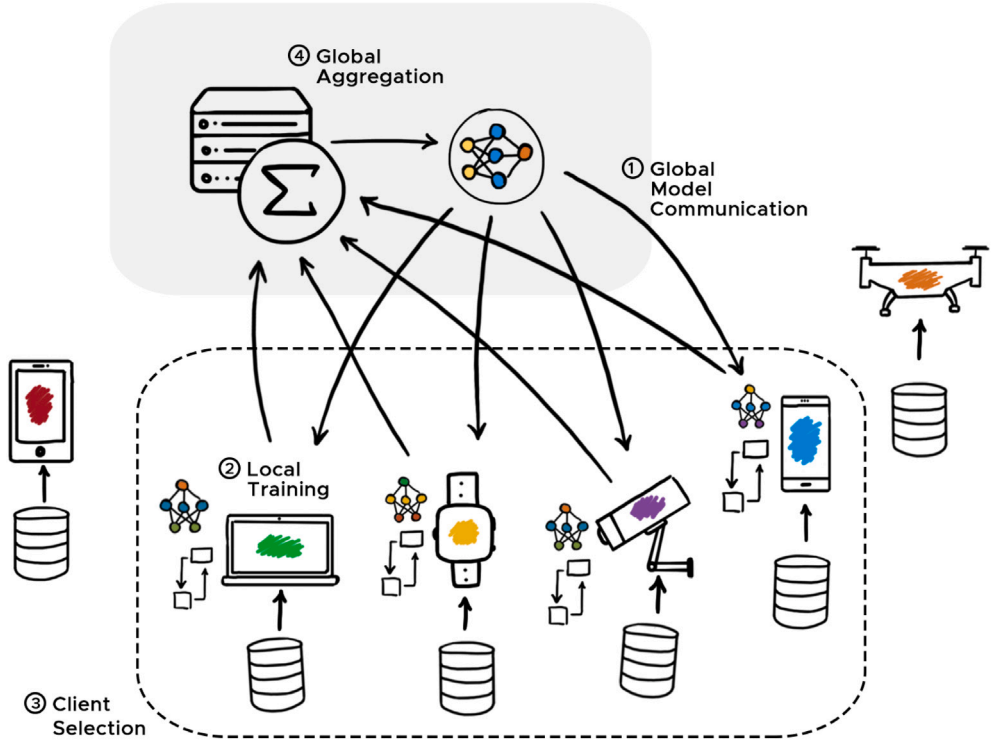


Fig. 2. Overview of the training process in FL.

other words, for a set of n number of example input–output pairs $\{x_i, y_i\}_{i=1}^{n}$, the model aims to find the optimal weight vector(s) w such that the loss function is minimized. A simple example could be seen in the loss function for a linear regression problem [20]:

$$f_i(w) = \frac{1}{2}(x_i^T w - y_i)^2, y_i \in \mathbb{R} \text{ and } x_i \in \mathbb{R}^d \tag{1}$$

Although loss functions are specific to each ML technique, the problem in most cases could be formulated as an optimization problem for a finite-sum of functions, as follows:

$$\min_{x \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{2}$$

In the context of FL, the training process that minimizes the loss function takes place locally through gradient descent before the weights w are communicated to the central server [8]. Assume an FL system with a fixed number of clients K, and each client is hosting a partition of the data n_k, indexed by the set of indexes $\mathcal{P}_k = \{1, 2, \ldots, i\}$. Upon receiving all local parameters, the global server aggregates w to update the new global model. Based on the aggregation mechanism the global objective is set. For instance, In FedAVG one of the most commonly used aggregation methods in FL [8], the loss function is updated to

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w) \tag{3}$$

Obviously, the problem becomes that of learning the aforementioned model distributively without exchanging of data.

2.2. Variation of Federated Learning infrastructures

Generally speaking, variation is an outcome of the development of any field of research, owing to diversity in thoughts and perspectives among researchers. Although FL is relatively young, it bore witness to an enormous amount of growth and evolution, giving birth to multiple variations and branches of the original concept. Generally, these variants can be categorized as follows:

i. **Centralized, decentralized and semi-decentralized Federated Learning:** Whereas the overview of FL given above assumed the existence of a central server (i.e. centralized Federated Learning), the development of a fully decentralized system for FL was suggested in multiple pieces of work [22]. The core concept is to achieve an aggregate of all the local updates by exchanging weights among clients, eliminating the need for a central server. In some implementations, the consensus schemes typical to blockchains motivated the incorporation of blockchains into FL. Kang et al. employed the Proof-of-Verifying

mechanism, inspired by the Delegated Proof-of-Stake algorithm, to secure and select clients for aggregation [23]. Miners verify the performance of each update from the clients and load the best few based on a specific threshold into a block, which is then broadcasted to other miners for the consensus process to take place. Clients download the new block and average out the parameters locally, signaling the end of the round. Other consensus mechanisms, such as Proof-of-Work [24,25] and Proof-of-Stack [26], were used in a similar manner to achieve secure decentralized learning. Another related variation is a hybrid of the two categories commonly called semi-decentralized FL. This configuration involves the organization of clients into clusters, where each cluster communicates its weight updates to a separate small-scaled server at the center of the cluster, often referred to as an edge server. The servers then improve the global model by means of aggregation in a decentralized fashion [27].

ii. **Cross-silo and cross-device Federated Learning:** Another way to categorize FL architectures is based on the scale of data and training capability of clients. Cross-silo in the context of FL refers to any system that utilizes silos as clients with large data and computational centers [28]. Due to the expensive nature of this structure, the number of clients is expected to be significantly smaller, with very reliable connectivity and processing power. Such variation is typical to large organizations, as implementations were seen in the medical [29] and agro-food sectors [30]. Comparatively, in cross-device FL the clients are represented by an extremely large number of devices with much smaller data storage capacity and computational power. Moreover, the connectivity and network reliability of these clients are typically low; therefore, selection of clients and error handling are vital. This scheme is often seen in systems that utilize mobile devices [31] and unmanned aerial vehicles [32].

iii. **Horizontal and vertical Federated Learning:** It is often the case that the partitioning of the training dataset in FL plays a role in differentiating systems from one another. The distinction is made based on the common space all clients' datasets share when data is partitioned. In the case of Horizontal FL, the feature space, which is often represented by the columns in a dataset table, is the same across all datasets, meaning that partitioning happened with horizontal cuts; across the sample space [33]. In Vertical FL, on the other hand, the clients share the same data samples but not the attribute space, which implies collaboration as opposed to the competitive nature of data privacy-preserving Horizontal FL. Furthermore, the difference in configurations means a difference in the number of clients; since the feature space is often significantly smaller than the sample space, having a large number of clients could be redundant [34]. Moreover, varying attribute spaces raise a need for unique ML models, making the nature of federation different, as only parts of the global model are trained locally, and the process of aggregation becomes that of combining the parts together as opposed to averaging full sets of weights, as is the case in Horizontal FL [34].

iv. **Single-layer and hierarchical Federated Leaning:** As has been alluded to before, typical applications of FL, especially those of cross-device configuration, host a very large number of clients, managed single-handedly by a singular central server. This could prove problematic for many reasons at that scale, hence why the introduction of some hierarchy into FL was deemed necessary. Hierarchical FL refers to an architecture that uses intermediate servers to mitigate a large amount of the traffic between the clients and the central server [35]. This is possible through performing the process of aggregation at each intermediate layer before the final aggregation at the center. Any number of intermediate layers could be added to the hierarchy; nevertheless, most implementations find one layer to be sufficient and it is often a layer of edge servers [36–38].

2.3. Aggregation algorithms

The heart of an FL system lies in its aggregation mechanism; through which multiple separately trained ML models are combined into one, that is capable of drawing inferences on each of the individual training datasets. Two of the model popular aggregation algorithms used in FL are:

i. **FedSGD** is often considered to be the base-line aggregation approach in FL. It involves each client calculating its own gradients $\nabla F_k(w_t)$ before averaging them at the center and updating the global model [8]. Hence, an update with an arbitrary learning rate η is written as

$$w_{t+1} \longleftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(w_t) \tag{4}$$

ii. **FedAVG** is a more communicationally efficient approach, rendition of FedSGD, that involves averaging the weight updates instead of the gradients, allowing the clients to perform multiple gradient steps before starting a communication round, hence achieving convergence at a faster rate, and a reduced number of communication rounds [8].

$$w_{t+1} \longleftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{5}$$

2.4. Role of Federated Learning in Internet of Things

The past years have witnessed stunningly rapid development and adoption of IoT technology, with smart things building a stronghold in the technological market. The fame and popularity of AI meant that its incorporation into IoT was only a matter of time. This merger gave birth to AIoT where AI methods such as DNN are leveraged at the cloud, fog, or edge to enhance the process of perception in IoT, with minimum latency. Since privacy is a very instrumental issue in IoT, the introduction of FL to

AIoT is a necessary step in the evolution of AIoT. As mentioned before, the training structure of cross-device FL is highly fitting for AIoT architectures, where the central server can take the role of the cloud, and the clients are taken by the fog and edge in a hierarchical topology. Nonetheless, edge and fog devices are often ill-equipped for the tasking process of training ML models, due to their limited computational prowess [15]. Furthermore, IoT networks often rely on the existing internet communication infrastructure which is highly underoptimized for FL. Hence, the issue of communication and computation efficiency is directly influential in the development and growth of the field of AIoT.

3. Communication efficiency in Federated Learning

Challenges such as improving model performance and handling low-quality data are not unique to FL, as a matter of fact, they are broad issues that need to be addressed in all branches of the discipline of ML. Enhancing the efficiency of communication, on the other hand, is a challenge centric only to FL, due to its dependency on data communication when exchanging parameters to preserve client privacy. If one were to replace the common centralized data servers with FL architectures, data communication would need to be efficient enough so as to be possible in the first place. Hence, the problem of communication efficiency is deemed to be one of the most significant core challenges for FL-oriented research.

Communication efficiency is said to have been achieved if data could be transmitted through a channel with minimum energy consumption. Generally speaking the energy needed for communicating a block of data of length $|D_{ij}|$ from i position to j is defined by

$$E = \frac{P_T |D_{ij}|}{C} \tag{6}$$

where P_T is the transmitted signal power and C is the channel capacity or the data rate of transfer. Additionally, The channel capacity is affected by the bandwidth of the channel and the signal-to-interference-plus-noise ratio (SINR). This paints a picture of the multitude of parameters that could be manipulated and methods that could be taken to achieve efficiency. Naturally, It follows that the solutions proposed for the improvement of communication in FL are diverse in their approach and methodology.

This section presents the current state of FL research in regard to communication efficiency, providing a systematic categorization of the myriad of methods suggested based on the specific challenge the works address. For ease of navigation, Fig. 3 shows our categorization of this research work and the respective subcategories based on the approach used in the solutions.

3.1. Client selection and dynamic scheduling

Exchanging model updates from and to the full list of participating clients contributes to the communication bottleneck during an FL training round. Random selection of a subset of clients is a viable solution, but that randomness may result in a lot of missed potential. In most FL implementations, the clients vary in design and capability, a diversity that extends to the quality of communication mediums. Choosing the clients that meet the most favorable communication conditions in each round should help achieve a higher average data rate, hence reducing the gap towards communication efficiency.

3.1.1. Stochastic client selection

Stochastic optimization is a process in which randomness is utilized to minimize or maximize a certain task [39]. This kind of algorithms is preferred over deterministic ones in cases where one might be dealing with high non-linearity, dimensionality or noise inherent to the problem. Chen et al. devised a client selection approach that arrives at the most optimal probability sampling for clients in each round of communication [40]. The work availed itself of the Stochastic aspect of Stochastic Gradient Descent (SGD) present in the unbiased estimation of the gradients of the loss function. By minimizing the variance between the estimated gradients of a subset and the full set of clients the optimal probability model for client selection could be achieved. The analysis estimated that this model could be found via the norms of the estimated gradients alone, which could be communicated to the server for computation with negligible communication cost. Further analysis to prove a guarantee of convergence was performed, and the experimentation results exhibited a very small decrease in accuracy from the full set of participants compared to uniform random sampling.

As is the case with FedAVG and many other FL systems, random client selection is performed in an unbiased fashion. Biased Stochastic client selection, in contrast, was suggested in [41], where the probability distribution for the process of selection is skewed based on the loss value generated by each client from the global model. In Cho et al.'s work, a comprehensive analysis of the nature of model convergence in FL was laid, concluding that skewing selection to accommodate more clients with large losses results in a boost in convergence speed, with the down-size being further distance from the global optimum represented in their analysis by a non-vanishing bias term. To handle this down-size, the Power-of-Choice algorithm was proposed, formulated based on a well-known load-balance strategy: power of d choices [42]. The algorithm simply selects a subset of d clients from which it randomly selects an active set of the clients, with those enjoying a higher global loss value being more likely to be selected. Their experiments show that the solution resulted in three times the convergence speed with an increase of 10% in testing accuracy.

FLOB is another framework that makes use of biased stochastic optimization to select clients in FL [43]. The essence of FLOB revolves around finding the best probability distribution that aids the server in selecting a near-optimal subset of clients that produce the lowest global loss, or as the paper describes it: lower-bound-near-optimal probability distribution. In comparison with FedAVG [8] and FedProx [44], FLOB shows greater improvement in inferring accuracy and loss, as well as a significant reduction in

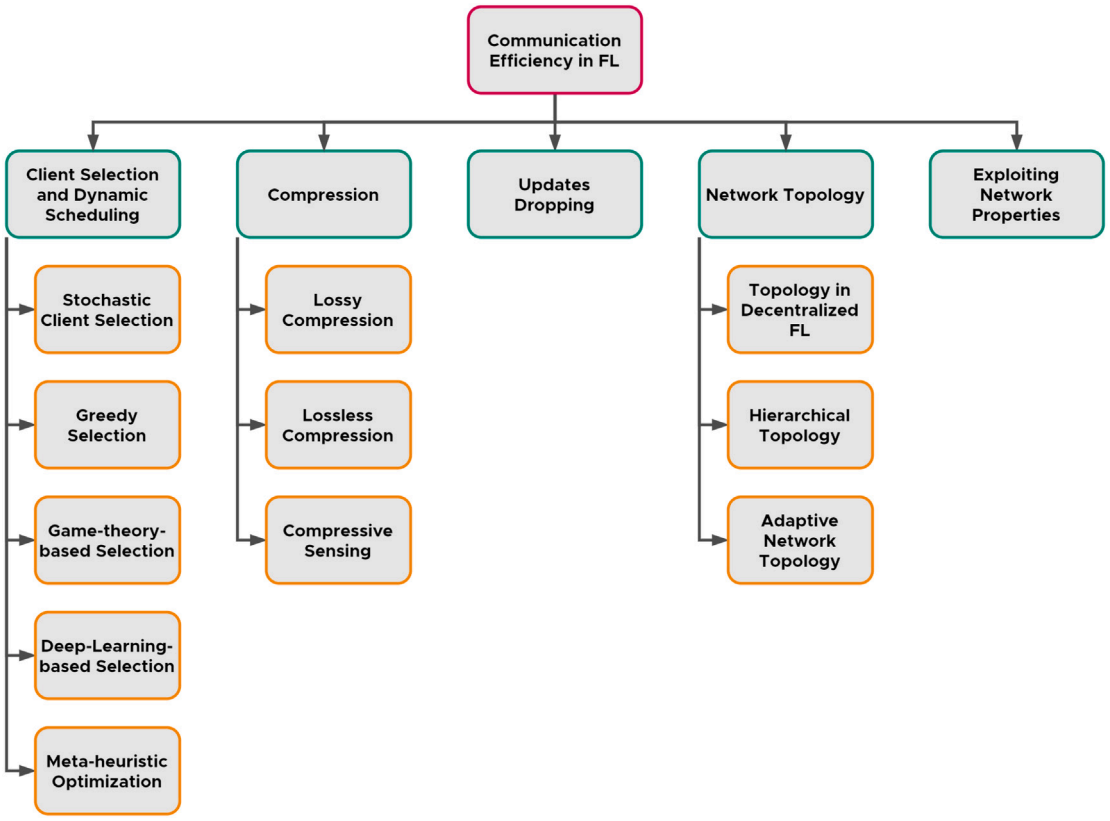


Fig. 3. Categorization of the literature on communication efficiency.

the number of communication rounds. Similarly, Chen et al. developed another biased client selector coined AdaFL, which assigns higher probability to clients with higher attention value, calculated based on the distance between the local updates tensor and the global one [45]. Additionally, the framework introduced an algorithm that dynamically arrives at the most suitable fraction of clients to be selected.

The client selection methods discussed thus far based their selection mechanism on the performance of the global model based on convergence, with little to no considerations for client communicational or computational failures or training biases. The research conducted by Huang et al. took these considerations in mind when formulating their Stochastic solution [46]. In order to achieve optimal selection with the noted constraints the problem is formulated as an Adversary Bandit, and is solved using the Multiple Play Exponential-weight algorithm for Exploration and Exploitation (Exp3) [47]. The objective of the algorithm is to maximize the reward gained from drawing a set of arms(clients) selected at a certain probability. In accordance with this method, Exp3 arrives at the most optimal probability allocation and a multinomial distribution is then employed to randomly select the active set of clients based on probability allocation. In comparison with other selection strategies, the proposed method produced double the convergence speed at a similar prediction accuracy.

The same approach of local resources being the basis for client selections was observed in [48], only with consideration for the future state of the resources. The algorithm, which was named Online Client sElection and bAndwidth allocatioN algorithm (OCEAN), leverages the current state of resources and a queue that keeps track of previous states of energy to effectively select the most efficient subset of clients for participation. Empirical experimentation depicted better model performance with minimal energy consumption, in contrast to other optimization methods.

3.1.2. Greedy selection

An optimization algorithm is considered greedy if it works with the goal of finding the best solution or set of solutions possible, that is to say, it does not accommodate any exceptions or leave any room for chance [49]. This subset of optimization algorithms is flexible, and it is natural to see them being employed for the selection of clients. Federated averaging with diverse client selection (DivFL) is a greedy approach to solving the issue proposed by Balakrishnan et al. in [50]. This study formulated the problem as a Facility location function [51], which is a versatile submodular mathematical function that produces excellent representations of a data space when maximized. A naive greedy algorithm was deployed to aid in selecting the best-limited subset that approximates the gradient space of the full client set; the best client is then chosen and added to the set followed by the second best and so on

until the desired number of clients is met. DivFL was simulated and tested on the famous FEMNIST dataset [52] and a synthetic dataset described in [44], and the performance during which was recorded in contrast with random sampling and the previously discussed Power-of-Choice [41], and it gave higher performance in terms of speed of convergence and classification accuracy in both tests.

Another work utilizing greedy algorithm for client selection is called Multi-Criteria Client Selection Model for Optimal IoT Federated Learning (FedMCCS), which considers multiple criteria when selecting the clients [53]. These criteria are training time, memory size, CPU's capability, and energy consumption during training. The paper formulates the problem of client selection as a submodular bi-maximization function subject to the constraints of Knapsack, which is maximized via a greedy-based heuristic algorithm. The steps taken towards the selection are as follows: the server sends a resource request to a random subset of clients, each of the clients provides information about their resources and sends them as a response, the response provided and a regression-based prediction on the resource utilization of each client are used to greedily pick the best clients for participation, and the rest of the training round is carried in a typical fashion. The authors contrasted their method against typical FedAVG [8] and FedCS [54], using the NSL-KDD public dataset [55]. Accuracy analysis showed that FedMCCS achieved higher convergence accuracy than both methods, achieving 80% within an 8 times less period of time. Further testing on the discarded rounds of communication, depicted only 45 rounds discarded during training when FedMCCS was tested, compared to FedAVG and FedCS at 686 and 667, respectively.

To ensure early convergence while avoiding communication overhead, Wang et al. [56], introduced Communication-Mitigated Federated Learning (CMFL); which selects a subset of clients with parameter updates that subscribe the most to the general trend of the global model. An update's relevance is inferred by matching the signs of each parameter in a local update to those of the global, based on which only clients that meet the relevance threshold communicate their updates for the round. When CMFL was utilized, model convergence seemed to be reached in about 13.97 times fewer communication rounds as compared to conventional FL. This rang true in the case of multi-task FL as well, where it even improved the testing accuracy on top of the expected communication cost.

3.1.3. Game theory-based selection

Game theory is a mathematical field of study concerned with situations in which rational players interdependently interact with each other based on a conflict of interest. In the auction mathematical game models in which players take the role of self-motivated bidders or auctioneers, players with the best bids are rewarded with a certain payoff [57]. In the context of FL, Le et al. attempted at formulating the problem of FL as an auction game, in which clients represent the bidders and the central server is the auctioneer, with the payoff being the selection of the client for participation in the round of communication [58]. The problem was defined as a minimization problem for a non-convex objective function that takes into consideration a measured lower bound of global epochs based on the local accuracy, an estimation of the total required energy for communication, and the total time needed it takes for the local parameter updates to be computed during training. Clients utilize the objective function to iteratively optimize it based on their available resources, and the bid is then made based on the sub-channel number, antenna number, local accuracy level and the training loss/cost. The auctioneer collects the bids and maximizes the social welfare of the system by selecting the best subset of clients by means of a greedy approximation algorithm. Resources at the chosen clients are optimized, parameters are sent to the server and each participating client receives their payment.

Zhang et al. designed a similar auction-based incentive mechanism to effectively select clients in FL setting [59]. Their contribution to the idea was by adding a reputation model based on the contribution of the clients and their trustability. The authors then improved on the framework in another study, by devising an "ex-post payment" incentive mechanism, that distributes payments after the task was performed based on the actual performance of the model, in contrast to the workers' (clients) claims during bidding [60]. Furthermore, they published yet another paper that enhances the scheme by applying it to an online setting where clients are free to participate and leave model training before or during tasks [61].

3.1.4. Deep learning-based selection

Research interest in the utility of Deep Learning models for solving complex optimization problems is not by any means recent, their robust and protean design nature makes them viable for highly noisy and non-convex problems. Zhang et al. proposed Deep Deterministic Policy Gradient (DDPG), which is a Deep Reinforcement Learning (DRL) algorithm, as a reliable method for improving communication by means of optimally selecting clients for global aggregation in FL [62]. The modeled cost function to be minimized took into account the communication cost, the training time, and the quality of training represented by the loss value. Typical to any Reinforcement Learning algorithm, it defines a reward to be given to the trained agent based on the global state of the federated environment and the action taken (the clients selected); the reward consists of the average cost value for the selection. A simulation was set up with 1000 clients, trained on both the Independent and Identically Distributed (IID) and the non-IID versions of the MNIST [63], Fashion MNIST [64], and CIFAR-10 datasets [65], and showed a significant decrease in communication rounds when compared to FedSGD [8].

Similar use of DRL for picking optimal sets of clients, coined FAVOR, was proposed by Wang et al. with a specific focus on Non-IID data, and utilizing Deep Q-learning (DQL) instead [66]. A dimensionally reduced tensor containing the weights of each client along sides the global weights defines the state space, while the action space reflects a size-specific set of top clients with the model trained to pick one at a time to iteratively fill the set. Based on a cumulatively discounted reward that measures the distance between the testing accuracy and a set desired accuracy, the model performs its actions and updates the state. The same datasets in [62] were deployed for evaluating the performance of the suggested FAVOR paradigm, where it exhibited a substantially reduced number of communication rounds, with a slight enhancement in accuracy contrasted with FedAVG [8].

In both [62,66], only one agent was trained at the server, which often results in a large action space that slows convergence. To address this issue Zhang et al. formulated a novel method coined Federated Multi-Agent Reinforcement Learning (FedMARL) [67]. FedMARL moves the RL model (agent) from the central server to the clients, by installing an agent at each client device tasked only with making a decision on the participation of that client in that communication round. Due to the very small action space, a simple MLP was deemed enough to construct the client, not to mention that it makes consideration for the low computation power of most client devices. The reward is given to each agent based on the mathematical formulation of the problem, as a maximization problem of a function of the accuracy, communication cost and computation latency. The state space on the other hand was modeled based on six terms: the local loss value, computational latency of local training, previous communication latency, current communication cost, and the index of the training iteration. Global model accuracy is sent to each agent to calculate the rewards, and so client selection improves with each communication round. FedMARL did not only acquire the overall best accuracy but also displayed 2.2 times improvement in communication cost and 1.7 times decrease in processing latency, when compared to 8 different state-of-the-art communication efficient FL schemes.

3.1.5. Meta-heuristic optimization

There often arise optimization problems to which exact solutions are either unobtainable or computationally highly expensive to calculate; metaheuristic algorithms are a class of non-problem-dependent optimization paradigms that address these particular issues by intelligently aiming for near-optimal solutions instead of optimal ones. Particle Swarm Optimization (PSO), is a well-researched method that falls under that class, and was deployed as a viable solution for client selection in [68]. FedPSO, the algorithm proposed in the work, does not select a subset of clients to send updates for aggregation, it instead picks only the one with the global best $gbest$ value, which is the minimum particle best $pbest$ value achieved by the client. $pbest$ is the output of the cost function generated during training based on the weight updates produced during particle swarm search. The client with the $gbest$ communicates its updates to the server, replacing the past global parameters and concluding the training step. The experiment conducted compared FedPSO with the stable FedAVG [8], both of which were trained on the CIFAR-10 [65], and MNIST datasets [63]. Superior accuracy and convergence speed were obtained by FedPSO, with similar communication cost to using five clients in FedAVG.

3.2. Updates compression

Adequate energy consumption could be realized with little consideration for a channel's capacity when the data exchanged is small in size. Compression is the process of reducing the size of a data block through various means to decrease energy consumption during transmission, and by extension, the time taken to receive the full block. In FL, compression is especially useful when it comes to dealing with DL models, as the size of updates communicated is often huge.

3.2.1. Lossy compression

Ideal effective compression paradigms do not exist, they must either compromise on computation cost or loss of information. Lossy compression is a scheme that compromises the latter. The work of Caldas et al. is an instance of this, suggesting a lossy compression scheme for data size reduction during communication [69]. The compression methodology consists of three stages, the first being basis transform performed on the flattened weight vector of the local model, so as to minimize quantization error. Secondly, uniformly random dropping of updates takes place, followed by the final stage of probabilistic quantization based on q-bit uniform quantization; that is to quantize values to either the minimum or maximum weight value in each of the q intervals based on a specified probability distribution. Experimental testing of the scheme demonstrated an approximate 8 times acceleration in communication.

A similar application of lossy compression can be observed in [70], proposing a sparsification-quantization-encoding model of compression, coined FedZip. Upon the beginning of compression, each parameter tensor is compressed separately, starting with sparsification based on the $Top - z$ algorithm [71], where a weight value is dropped only if it is less than the zth top weight parameter. The dithered tensors are then quantized via the k-means algorithm, where the clustered parameters take the values of their centroids. Moreover, communication is further facilitated by applying Huffman Encoding [72]; it is worthy of noting that this step is effective due to the cluster-based quantization performed previously, which produces values with high frequency, hence shortening the encoded output. Compared to FedAVG [8] and FedSGD, FedZip [70] obtained a slightly less testing accuracy average, with an upper bound of 1085 times the rate of compression.

A similar pattern to the dithering-quantization-encoding paradigm of compression seen in [70] is echoed in some other research work, with varying algorithms and results. For instance, Shlezinger et al. demonstrated a similar scheme with lattice quantization instead [73]. Sattler et al. in their Sparse Ternary Compression (STC) method [74], made use of top-k as their sparsification algorithm, quantized the rest of the values to ternary matrices of both the positive and negative averages [75], and encoded them by means of Golomb code [76]. In another study, a federated trained ternary quantization (FTTQ) was introduced in the same fashion saving for the encoder [77]. The study dithered values based on an adaptive threshold calculated in accordance with the sparsity of each normalized weight layer. Following sparsification, a trained ternary quantization (TTQ) method utilized the same threshold to quantize the remaining parameters [78]. Interestingly enough, in this work compression is bi-directional, only the global weight tensor is compressed as long as the drop in performance remains above 3%. Another bi-directional compression framework was developed by Zheng et al. in [79], where a scale-round-limit approach was explored in the uplink, and Layered Quantization for the downlink. Moreover, besides the aforementioned work, there exists plenty of other studies that relied on similar compression schemes in FL [80–85].

Compression through sketches, which is a group of dimensionality reduction methods is also widely adopted for facilitating communication in FL. FetchSGD is a sketch-based compression approach developed by Rothchild et al. in [86] utilizing Count Sketch [87] with momentum and error accumulation to ensure adequate recovery. Count sketch is also the building block of SKETCHED-SGD, a similar paradigm to FetchSGD with minor differences [88]. It was, as well, utilized in [89] with additional privacy preservation considerations. In all these implementations, the algorithm seemed to provide impressive performance in terms of compression rate and model performance.

3.2.2. Lossless compression

Lossless compression refers to a form of compression where no loss of information occurs, with the trade-off being a lower compression rate and/or higher computational cost compared to its lossy counterpart. PTSP, the compression algorithm used in [90], is a prime example of lossless compression, visible in the complete recovery of the weight tensors communicated in the network. This is achieved through neuron pruning and shuffling, the former conducted through merging an arbitrary number of weight values according to a constructed distance matrix, where pairs with the minimum values are merged. As for shuffling, it is performed between individual randomly picked weights iteratively for a pre-fixed number of times, keeping a record of all shuffles in the form of index pairs.

In [91] a lossless yet computationally inexpensive quantization scheme was devised for efficient FL. Instead of a commonly used linear quantization function, the authors opted for a non-linear cosine-based model instead, where weights and gradients are quantized into angles calculated by taking the $arccos$ of each parameter value in the flattened vector over its norm. To avoid wastage during compression a quantization bound is computed for precession. It was finally modeled into a probability function to avoid quantization bias. The full parameter vector can be recovered by reversing the process with help of the bound and norm values, sent on communication. The lossless nature of compression resulted in the maintenance of high accuracy compared to other linear compression paradigms, observed in the experimentation section of the paper, all while producing a compression rate of about 1000 times.

3.2.3. Compressive sensing

Compressive Sensing (CS) refers to a sensing scheme in which perceived signals are compressed at the time of sensing, which allows for reliable sampling at a lower rate than the Nyquist rate [92]. This is only plausible if the sensed signal is sparse and compressible, meaning a signal that contains mostly zero elements and its sorted elements tend towards polynomial decay. Note that this is applicable to the signal in any of its domains.

Since CS could only be used when dealing with sparse signals, Jeon et al. started their implementation of CS with the design of a signal transmission scheme that ensures sparsity in the communicated gradients [93]. This strategy involves the construction of random permutation matrices applied to the signal upon transmission, and reverts back upon reception, to ensure the preservation of sparsity in case the gradients had similar distribution. When the compressed signal is sent, the server iteratively reconstructs the gradients with guidance from the linear minimum mean square error (LMMSE) obtained for efficiency of computation via statistical information taken from and an approximation of the gradients collected at the client side. This method was compared to the other reconstruction algorithms that rely on LMMSE through experimentation conducted on the MNIST dataset [63], and showed a significant decrease in computational complexity, and closer classification accuracy to centralized ML.

Compressive sensing was shown to be integrable to the FL model in several other research studies, one of which was authored by Li et al. [94]. In their implementation, CS-FL and 1-bit CS-FL were produced and compared against several commonly used FL schemes. The work was divided into two phases, the first of which starts with the sparsification of the weight vector, with sparsified values stored in a residual vector. A random measurement matrix compresses the vector which is sent to the server for averaging. The global compressed measurement is then sent back to the clients for reconstruction via the iterative hard thresholding algorithm (IHT) [95], and updates to the global model take place locally, signaling the start of the second phase. Training occurs at the client side for a specific number of iterations before the residual vector is added back into the vector and quantized for communication. The server averages the quantized 1-bit vector and sends it back to the clients for the last update for the weights completing a full training cycle. The differences in 1-bit CS-FL from the standard method lie in the quantization of the compressed measurements in the first phase and the subsequent use of the Binary Iterative Hard Thresholding (BIHT) algorithm as opposed to IHT. Simulation proved that both algorithms show better convergence rates, higher testing accuracy, and lower communication cost as compared to FL-STC [74], SignSGD [96], and FedAVG [8], when tested on the MNIST [63] and Fashion-MNIST [64] datasets.

Quantized compressed sensing (QCS) is a variation of CS that quantizes the to-be-transmitted compressed signals for further reduction in data transmission. Oh et al. adapted the algorithm to FL systems by dividing the local updates into blocks and performing the compression scheme block-wise [97]. Typically, a random projection matrix is used to compress the block vectors, the result of which is then quantized using Q-bit scalar quantization. The produced signal is then transmitted to the server. Similar to the method seen in [93] the signal is reconstructed iteratively based on the minimum mean square error (MMSE). The algorithm explored for reconstruction was expectation maximization generalized approximate message passing (EM-GAMP) [98], where it approximates the signal by modeling each quantized vector through a Bernoulli Gaussian-mixture distribution. The testing results of the method demonstrated a near lossless reconstruction of the gradients, all while sizably reducing the communication cost.

Other adaptations of the same compression scheme seen in QCS can be noted with changes in either the recovery methods or the quantization method. An instance of the former is QIHT seen in [99] that recovers the quantized compressed signal using IHT (adapted for quantization). As for the latter, it can be observed in Abdi et al. exploration of a dithered quantization for CS (recovered through an unbiased and an MMSE-based estimator) [100] and the vector quantization method developed by Oh et al. [101].

3.3. Updates dropping

Stochastically dropping neurons out when training a neural network is a common technique utilized to overcome the problem of overfitting. To alleviate the communication overhead in FL, a similar concept could be employed. Just as is the case in compression, dropping some portion of the updates could help reduce the amount of data communicated and thereby enhance efficiency and decrease communication time. The methodology for how to decide which neurons to drop out differs among the literature.

The compression aspect of [69] has been discussed in the previous section, nonetheless, the authors extended their model of communication efficient FL to accommodate a dropout mechanism as well, coined Federated Dropout (FD); inspired by the widely used dropout technique. The essential idea behind this scheme is to omit a fixed number of uniformly random activations in case of a Multilayer Perceptron (MLP) or filters in case of Convolutional Neural Network (CNN), hence producing a sub-model that is sent to the client for training. This not only reduces the communication cost due to the drop in communicated data, but it as well cuts down on the computation cost.

Adaptive Federated Dropout (AFD) [102] was a dropout method developed by Bouacida et al. built on the work of [69]. It sought to fill in the gaps left by its predecessor: the random nature of dropout which does not account for the unique configuration of the neural network, and the lack of attention to memory-based networks such as RNNs. The authors proposed to introduce a constructed activation score map that decides on the communicated activations based on relevance. Similar to [69] filters are replaced by the activations in CNNs. Furthermore, considerations for RNNs were taken by only dropping connections of non-recurrent nature. Utilizing this score map concept, two schemes were developed: Multi- and single-model AFD. The former constructs multiple score maps to send unique sub-models to each client, whereas the latter constructs one map to be sent to all clients. Conducted Experimentation demonstrated improved overall accuracy, convergence time, and compression rate to traditional FL and traditional FD.

In an attempt to minimize the effects of parameters dropping on the accuracy during the training process, Zhu et al. suggested the formulation of a bi-objective problem for the optimization of both the global model accuracy and the overall communication cost induced in the process [103]. The modeled problem was solved using the genetic algorithm: non-dominated sorting genetic algorithm II (NSGA-II) [104], where the chromosomes consisted of the model's hyperparameters (selected for improving the accuracy) and the parameters of the modified sparse evolutionary training (SET) algorithm, employed to sparsify DL models with minimal effect on inference efficiency, hence compressing them by dropping a fraction of the weights. The chromosomes were encoded in a hybrid form of binary and floating points. For experimental validation, an MLP and a CNN were prepared and tested on the IID and Non-IID versions of the MNIST datasets [63]. The numerical results exhibited a sizable decrease in model connections with a marginal reduction in testing accuracy, communicating only around 10% to 12% of the total model updates.

Contrary to the work that reduces the number of updates by dropping them directly from the update matrix, Wu et al. suggested reducing the size of the communicated parameters by means of knowledge distillation, which is the transfer of knowledge from a large model to a smaller one [105]. This approach (FedKD) trains a large teacher model and a smaller student model on the local data and only the student model is communicated to the server after knowledge was distilled between them. The communicated gradients at the server are aggregated and sent back to each client where the updates student model transfers its knowledge back to the teacher model, which is in turn used for inferring on new data samples. Considering the additional computation cost incurred, this solution targets cross-silo applications only. The updates upon communication are further compressed through a singular value decomposition-based approach. Comparison against the state-of-the-art approaches, including [69], showed a significant reduction in communication cost by orders of magnitude, with marginal reduction in training quality.

Asynchronous model updating is an FL method that reduces communication cost by communicating the updated parameters of deep layers less frequently than shallow ones [106]. The rationale behind this is that general data features are learned at the shallower parts of the model, hence holding a higher impact on convergence. Besides, the parameters associated with said shallow layers are often smaller in size, which reduces communication significantly. To ensure fast convergence, a state-of-the-art method for server aggregation was also employed. The framework was tested on two different datasets with different DNN architectures, and instrumentally less communication cost was observed.

3.4. Network topology refactoring

The configuration of connections and devices in a network is commonly called network topology. Intuitively, different topologies should incur different communication costs between communicating elements. Building upon this concept, recent work was dedicated to studying network topology as a possible area of contribution towards efficient communication in FL.

3.4.1. Decentralized Federated Learning

While optimizing network topology is deemed secondary in centralized FL, it is extremely essential in its decentralized counterpart, due to the lack of a central server. Segmented Gossip Aggregation (SGA) was proposed in [107] to optimize communication in decentralized FL. It is inspired by the gossip topology, which has been widely used in distributed optimization. In this method the clients act as workers that "gossip" segments of their updates to some other clients for aggregation, hence propagating the averaged model across the network. In a similar study, Wang et al. used the consistent hashing algorithm [108], to construct a ring topology that elevates half of the communication pressure off of clients [109], while maintaining the same data rate as [107]. Both topological configurations showed improvement in convergence, testing accuracy and channel capacity as compared to the fully connected topology model.

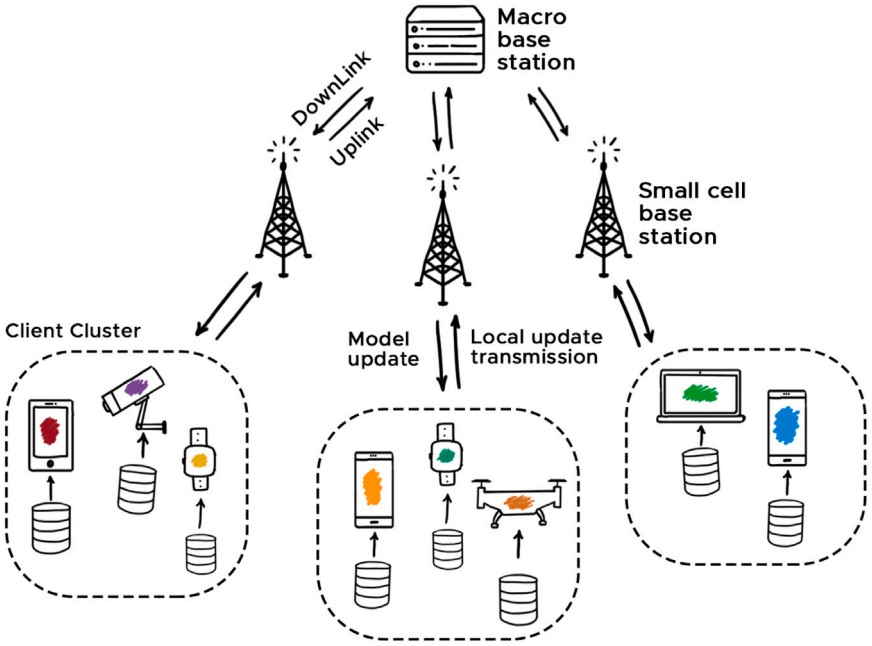


Fig. 4. Example of a Hierarchical Federated Learning architecture.

Another work that leverages topology in decentralized FL to improve communication and convergence was the D-Clique topology by Bellet et al. [110]. Although the main intention behind designing this topology was to cope with heterogeneity in data, it proved to be effective at improving network communication as well. D-Cliques are built on the idea of having fully connected subsets of clients, each of which is called a clique [111]. These cliques are by turn connected based on an inter-cliqued topology, for which four choices were suggested and tested during experimentation: fractal, ring, fully-connected and smallworld. This particular topology exhibited a huge reduction in communication cost, keeping only 96% edge connections, that communicated around 96% less times than the traditional fully connected scheme.

3.4.2. Hierarchical topology

Hierarchical FL follows the tree topology and was suggested as an approach to reduce communication latency, and overcome the bandwidth constraint of the server by using small cell base stations (SBSs) as a mediatory layer for partial aggregation. Proximate local clients aggregated their updates at the SBSs before sending their averaged updates to the server, which is the macro base station (MBS) in the context of Hierarchical FL, for further aggregation, as illustrated in Fig. 4. In a modern IoT setting, the MBS could be replaced with a cloud-based server and the SBS by fog or edge servers, as seen in [36].

Several studies had attempted to improve communication in this topology even further. For example, Abdellatif et al. attempted to enhance the assignments of clients and communication resource allocation, to boost the convergence rate and reduce latency [35]. In their work, the problem of Hierarchical FL was modeled as a Kullback–Leibler divergence (KLD) minimization problem, which was solved analytically. On the other hand, He Yang suggested clustering end user devices based on the KLD and entropy of the local datasets through K-means, and assigning each cluster to a separate SBS [112]. Further enhancement of communication latency in FL systems with hierarchical structure was suggested in [113], where over-the-air aggregation, which will be discussed in Section 3.5, was employed at both the SBS and the MBS.

3.4.3. Adaptive network topology

The common star topology seen in the traditional FL architecture is ideal for its cross-device configuration; however, the data silos that cross-silo FL systems are built on enjoy just as great or even greater channel capacity than the central server, which causes a great deal of inefficiency, prominently observed in the form of a bottleneck. Marfoq et al.'s solution was a network non-specific topology optimized for maximum throughput, which is the number of communications performed within a specific duration [114]. This was achieved by modeling the FL network as a directed graph called a connectivity graph, which simplifies the actual communication infrastructure by concentrating on the silos and the connections between them. The paradigm suggested is tasked with finding the best topological subgraph that optimizes the system's throughput, modeled by means of the theory of max-plus linear systems. Moreover, decentralized periodic averaging stochastic gradient descent (DPASGD) was employed as a replacement for the averaging model introduced in FedAVG, which improves communication even further [115]. Compared to the typical star topology, experimentation showed about 9 times improvement in data rate, and about 1.5 folds the improvement compared to the state-of-the-art MATCHA topology used for distributed model training [116].

3.5. Exploiting communication channel properties

Some inherent properties of communication channels could be exploited to alleviate some issues that arise in a particular communication setting. As a solution to the communication overhead in FL over the wireless medium, exploiting the wave superposition property of multi-access channels to perform aggregation "over-the-air" (OTA) was proposed by multiple studies including [117]. To ensure minimal aggregation error, and optimal selection of client devices, the problem was modeled as an optimization problem with sparsity in its objective function and a low-rank constraint. Furthermore, the problem was solved using difference-of-convex-functions programming and its associated algorithm. The simulation demonstrated improved communication with near identical training prediction accuracy and loss to conventional FL.

Another research paper by Zhu et al. addressed the issue of deployment of OTA-based schemes in modern digital wireless systems by digitalizing the broadband OTA process [118]. This was possible through quantization and modulation of weights at the edge, using one-bit gradient quantization and Quadrature Amplitude Modulation (QAM), respectively. The compressed updates are thereafter decoded at the server through over-the-air majority voting. Sery et al.'s variation of OTA aggregation, on the other hand, focused on mitigating the effects of noise produced during aggregation, especially when dealing with heterogeneous data, through precoding and scaling of communicated updates [119]. Yet another research work was dubbed Unit-Modulus OTA computation (UMAirComp), which reduces both the algorithmic complexity of optimizing aggregation and the cost of implementing RF chains in multiple-input multiple-output FL [120]. Other similar OTA aggregation approaches were also proposed in [121,122]

4. Computation efficiency in Federated Learning

Despite the wide variety of ML algorithms and techniques, they are considerably computationally complex. This is especially true in the case of DL, where complexity is proportional to the size of the neural network. In traditional centralized ML, the issue is widely mitigated by boosting the computational capabilities of the computing hardware, particularly during the training phase. Nevertheless, such an option is not always available for cross-device FL, where the utility of devices designed with little to no consideration for ML training is fundamental. This is especially true in the case of IoT applications, and mobile crowdsourcing, which are the two most common uses of cross-device FL. Thus, ensuring computational efficiency is of the essence if FL is to be implemented.

Efficiency in the context of computation is often understood in terms of the time and energy it takes to perform certain computational jobs. This is similar to the discussion on communication in the previous section in which physical parameters can be manipulated to improve it. However, computation can also be enhanced by augmenting the algorithms involved in these jobs themselves. This explains the wide use of BachmannLandau notations as measures of the computational complexity of algorithms [123]. Since the issue of computation is this multifaceted, it is naturally expected that the literature on the topic would suggest varying solutions with a focus on different aspects of the problem. This prompted the division of this section into the subsections seen in Fig. 5.

The present section brings forth a summarized view of the research conducted thus far on tackling the issue of computation inefficiency in FL. The reader should note that while communication and computation efficiency are interlinked especially when it comes to FL, the problem of computation is its own entity that deserves focused attention. Nonetheless, this interlinked nature explains why some papers are discussed in both sections of this paper. However, such discussions will be limited to computation, as was similarly done in the previous section covering communication solely.

4.1. Resources allocation optimization

The computational capabilities of a device are limited by the device's hardware resources. The recent accelerated development of computational chips and their architecture made resource under-utility in the day-to-day usage of personal devices unnoticeable. Nonetheless, in the case of the computationally heavy FL, these limits start to pose a significant threat to training efficiency, therefore, optimal utilization is a necessity. Proper computational resource allocation in FL was a topic addressed by a sizable amount of research work, which was hereby summarized, according to the method of optimization.

4.1.1. Analytical methods of optimization

In any typical application of ML, the frequency of the CPU's clock directly affects the amount of energy consumed during the training of the modal locally [124]. To optimize energy consumption for FL over wireless networks, Tran et al. proposed FEDL which allocates the required frequency to minimize the energy consumption and computation time for each of the participating clients [125]. FEDL mathematically categorizes the client device into one of three categories: devices that must run at maximum frequency to perform training dubbed "bottleneck", devices that can train the model on its minimum frequency and finish within the assigned time called "strong" and devices whose optimal frequency is within the bounds of the feasible sets in which case the frequency is weighted such that it the minimum amount of training time is achieved with minimal energy consumption possible.

Yang et al. proposed a framework that consists of multiple algorithms for optimized resource allocation for both communicational and computation efficiency [126]. The resources targeted in the work involved the CPU's frequency, the bandwidth of communication, the transmission power, the communication time, and the accuracy of the model, all of which were controlled so as to enhance the rate of energy preservation. The methodology suggested involves two stages, an initial solution is first found through a bisection method based on which the accuracy and communication time were calculated via the Dinkelbach method, by fixing the rest of the resources. The second stage then involves solving for the rest of the values based on the solution of the first stage. The two stages are then iterated over until convergence. The experimentation conducted compared the proposed solution to the traditional approach for FL and demonstrated about 59.5% improvement in energy consumption.

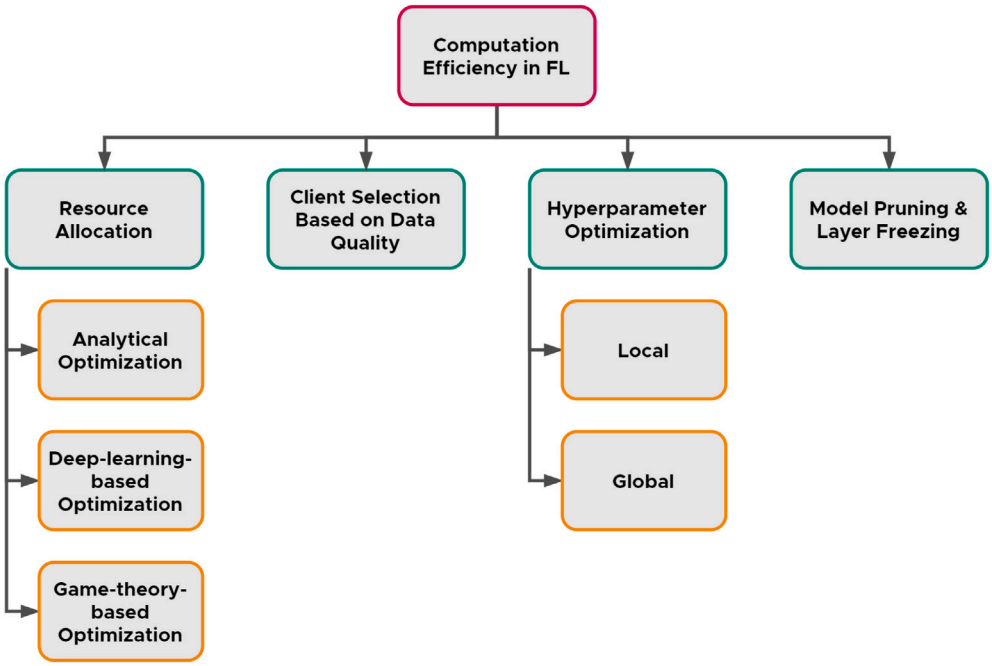


Fig. 5. Categorization of the literature on computation efficiency.

4.1.2. Deep learning-based optimization

While static, direct, mathematical optimization is feasible, as evident by the attempts seen in [125,126], it could be deemed impractical in networks with unstable client connections, where exact mathematical modeling proves difficult. Consequently, Zhan et al. overcame this shortcoming through DRL. The experience-based nature of DRL models ensures a near-optimal solution all while removing the need for a rigorous mathematical model that describes the quality of network connections. The work employed Proximal Policy Optimization (PPO) DRL algorithm with a state space that consists of past measurements of the bandwidths of each client device. Similar to [125], the optimized parameter was the frequency of the CPUs, and hence it was set as the action space. Simulated experiments showed significant improvement in reducing the computational cost for the overall system compared to traditional FedAVG and the method proposed by Tran et al. [125]. Similar methodology was adopted in [127] for networks that consist of mobile devices, only DQL was employed for decision making instead.

Nguyen et al. [128] proposed, an implementation of a DQL model to optimize the resource allocation of both computational and communicational resources in a mobile FL setting. Regarding computation, the issue of limited battery capacity in mobile client devices was the main concern, solved by means of a power beacon that recharges the devices remotely. To avoid the energy cost imposed by naive charging, the DQL model was employed to optimize the charging process. The action space, hence, was set up as the amount of energy assigned to charge each device in the network. The state of the model, on the other hand, involved the current energy state of the device, the quality of its communication channel, and its position pertaining to the range of communication of the central server. The DQL model was trained and tested against other resource allocation methods namely RL, greedy and random allocation, where it showed superior performance.

4.1.3. Game theory-based optimization

Game theory was not only proposed to improve communication efficiency as discussed in Section 3 but it was also proposed by some research work to improve the allocation of clients' computational resources instead. Sarikaya et al.'s work is a prominent example of this, where the Stackelberg game mode (known as well by the Stackelberg leadership model) was explored for the task [129]. The leadership role in the game is naturally assigned to the server, where it sets its budget, and the clients (followers), motivated by the potential revenue, negotiate about the optimal CPU power to reduce training latency and achieve Stackelberg equilibrium. Moreover, the game was split into two separate subgames; lower-level and upper-level, to facilitate analytical optimization. Empirical testing demonstrated great contribution to reducing training time delay and discussed important trade-offs associated with the parameters of the framework.

The crowdsourcing framework presented in [130] demonstrates a similar two-stage Stackelberg leadership model for resource allocation. The difference is in the proposed analytical solution that finds Stackelberg equilibria in linear time, as well as the addition of an admission control scheme to further enhance communication efficiency and ensure adequate weight update quality. Moreover, it is important to mention that these are not the only instances in which the mechanism was observed for FL; other studies were conducted with variations of the same game employed for the same task [131–133].

4.2. Client selection based on data quality

In Section 3 of this paper, the concept of client selection for communication efficiency was discussed. In these solutions, the selection process was based on the quality of the weight updates produced by clients during local training. This technique often gives rise to a tradeoff between communication and computation, as it results in discarded expensive computations performed by unselected clients. To overcome this tradeoff, client selection based on data quality was suggested.

Data-Quality-based Scheduling (DQS) was proposed by Taik et al. to reduce the computational cost associated with gradient-based client selection in federated edge learning [134]. Measurement of data quality in the study was induced based on the client's reputation and an index for the diversity of the data it holds. This measure was set to be maximized in an optimization problem model, solved via a greedy knapsack algorithm that enjoys both low complexity and high computation speed. Experimentation conducted by the authors exhibited sizable improvement in efficiency, while providing protection against data poisoning attacks known to affect FL systems.

Another attempt at avoiding unnecessary expensive computation through assessment of data quality was seen in [135]. The work formulates an irrelevance score based on three expressions; volume of data, class imbalance, and the non-IID nature of data. Upon calculation of the score, a sampling algorithm is deployed to produce a probability distribution for client selection based on three different sign-based pools. A certain number of clients is selected from each pool with a preference for those clients closest to a score of zero. An experiment was performed on 6 different datasets, and the sampling algorithm was observed to have achieved up to 80% faster convergence when compared to FedAVG [8] and other recent methods of client selection.

Although picking clients with the highest quality data is an effective way of improving the efficiency of computation all while improving the process of training, it is quite a rare occupancy to have data distributions with perfectly relevant data instances. Hence why Nagalapatti et al. suggested moving the process of selection towards individual data points instead [136]. In the paper, Federated Learning with Relevant Data (FLRD) was introduced to ensure the absence of irrelevant data. At the heart of the framework, an RL model named Relevant Data Selector (RDS) is trained on local data, with feedback from the global model, to assign relevance scores and select data points for training in each communication round. FLRD showed an observable boost in prediction performance, and faster convergence when compared to conventional FL and other recently developed FL frameworks in a simulated experiment.

4.3. Hyperparameter optimization

The parameters responsible for controlling the training process of any ML model are often referred to as hyperparameters. Varying these hyperparameters affects learning in a myriad of ways and randomly assigning them runs the risk of yielding unfavorable performance. Whereas the main goal in central ML is to boost the accuracy of the model while minimizing the convergence time, it is a different case in FL. The computational resources in federated training are often scarce with higher constraints. Many of these hyperparameters impact the computational cost of training immensely [137]. Having a very low learning rate, for example, results in prolonged training which significantly increases the number of computations. Another instance is the number of hidden layers and neurons in DNNs, which is proportional to the computational burden endured per training round. Selecting inadequate hyperparameters results in the wastage of valuable computations leading to a great deal of inefficiency. Consequently, this section was prepared to discuss the latest findings on the optimization of hyperparameters in FL. It should be noted that some of the work discussed does not target computational efficiency specifically, however, their unintentional contribution to the efficiency of FL was enough of a reason to be included.

4.3.1. Local hyperparameter optimization

The distinction between local and global hyperparameters in this section refers to the typical ML hyperparameters and the federated parameters that govern the communication and aggregation process in FL, respectively. The challenge of tuning the local hyperparameters takes a unique form in the case of FL, mainly due to the heterogeneity in client devices, their computational capabilities, and their local data. Setting the hyperparameters naively slows the process of global convergence, and in some cases enlarges the variance of weight updates which incurs a sizable amount of computational waste. One of the studies that sought to find a solution for the issue was [103] discussed in Section 3.4 for its contribution to improving communication efficiency in FL. The paper discussed the use of NSGA-II, a genetic algorithm, to tune the hyperparameters of the trained model, considering the learning rate, size of filters, and the number of nodes in each neural layer. The results showed a 10% improvement in convergence speed measured in communication rounds.

Another piece of work that explored a metaheuristic algorithm for hyperparameter optimization was conducted by Qolomany et al. where PSO was selected, for its effectiveness and low computational complexity, as the method for optimization [138]. The paper concentrated on industrial IoT and smart cities when implementing the algorithm, particularly testing its framework for traffic flow and maintenance need prediction (using the City Pulse EU FP7 and Microsoft Azure Intelligence Gallery's predictive maintenance dataset, respectively). The results were compared to those produced by GridSearchCV, where it was shown that PSO requires 98% less training before convergence.

Federated Loss SuRface Aggregation (FLoRA) is a hyperparameter optimization technique developed particularly for FL by Zhou et al. [139]. The framework operates by first finding the best local hyperparameters based on a single local training epoch by means of adaptive optimization (Bayesian Optimization). The selected hyperparameters are aggregated at the server via one of the four suggested methods of loss surface aggregation, based on which the best set is chosen and distributed back among the

client devices. To validate the proposed methodology, simulated experiments were conducted, where Boosted Decision Trees were trained and tested on seven different datasets. all four methods of aggregation were compared to the default parameters set by Scikit Learn the implementation library [140] and depicted noticeable improvement in terms of the amount of computation needed before convergence and accuracy.

Perhaps an obvious shortcoming that could be noted in [138] was its lack of consideration for the diversity of client devices and data when tuning the hyperparameters. Genetic clustered FL (CFL) was an algorithm developed particularly to address that diversity, by grouping clients into clusters, and optimizing the hyperparameters in models for each cluster [141]. A list of learning rates is declared and a random subset is sent to each client to be trained for one epoch, where the clients respond by sending back the learning rate with the minimum training error. Based on these learning rates the clients are clustered via Density-based spatial clustering of applications with noise (DBSCAN) [142], chosen for its dynamic number of clusters. Lastly, a typical genetic algorithm tunes the hyperparameters clusterwise. Genetic CFL was tested on MNIST [63] and CIFAR-10 [65] and showed a significant increase in both performance and convergence speed, achieving an accuracy of 99.79% and 76.88% respectively for both datasets.

In most implementations of FL the number of local epochs or training iterations is fixed for all clients, which Li et al. argued that it causes great inefficiency when dealing with device and data heterogeneity [44]. Differences in hardware specification and degree to which the data is Non-IID result in inconsistent local training time, which leads to slower convergence and waste of computational resources. Their solution was FedProx which is a framework that allows clients with a lower reservoir of computational resources to perform fewer local epochs before sending updates to the server. However, to avoid predictive divergence in the global model as a result, an addition of a proximal term to the local weight optimization problem was necessary (making each local weight update an inexact solution to the problem). implementation of FedProx displayed a faster convergence rate and an average improvement of 22% predictive accuracy. Note that the same suggestion of varying local epochs was proposed in [143], with the difference being the utilization of a DNN for deciding local epochs beforehand.

4.3.2. Global hyperparameter optimization

The convergence analysis conducted on FL recently by Wang et al. in [144] demonstrated a correlation between convergence speed and the frequency at which global communication rounds take place. The methodology involved analyzing local training in terms of elapsed real-time and server-based averaging of gradients to optimize the frequency of server averaging. The results of said analysis demonstrated that communication between the clients and the server should be incremented periodically to ensure optimal convergence speed. This finding was the basis of their adaptive communication strategy (AdaComm), which divides training time into fixed periods, in each of which communication frequency is decided analytically. This method reduced training time by three folds when tested on the CIFAR10 and CIFAR100 databases [65], via the DL models ResNet50 [145] and VGG16 [146].

4.4. Model pruning and layer freezing

When dealing with DL models, it is at times the case that not all neurons in the network are utilized for learning, resulting in a number of parameters that contribute little to nothing to the prediction process. The pruning of said neurons can save valuable computational resources with minimal effects on the accuracy of inference. This concept was applied to FL in a framework named PruneFL [147]. The framework prunes DL models by means of an adaptive algorithm in two stages. First the device with the best resources is utilized for the initial pruning based on its local data in an iterative manner until convergence. Then, the server picks up the task further pruning the model based on the parameters obtained during each training round from all local clients. To verify the effectiveness of the introduced framework, the authors tested it by incorporating four different NN architectures on four different datasets. It has been observed that PruneFL seemed to achieve convergence at less than a third of the time it takes conventional FL and other state-of-the-art pruning methods, requiring far fewer floating point operations per second (FLOPS) while maintaining almost the same inferring accuracy.

HeteroFL is an FL framework formulated by Diao et al. [148] that employs a similar concept to model pruning. However, instead of selectively dropping irrelevant neurons from the trained model, HeteroFL centers around the notion that large DNN models with wide layers still produce sufficient prediction accuracy upon width reduction. Accordingly, an adaptive algorithm was developed to assign subsets of the model with narrower layers to each client based on its computational abilities. Furthermore, the framework sought to address two additional issues; namely the cost of the statistical information tracked by Batch Normalization layers, and the divergence of parameter updates observed when models are trained with layer widths. The former was solved by statically normalizing mini-batches and the latter through a scaler layer that scales learned features similarly to dropout layers. A simulated experiment that consisted of three different DNN models trained on different datasets showed that HeteroFL achieved almost similar global prediction accuracy to that of FedAVG [8] with far fewer learned parameters and FLOPS at high scaling ratios.

While model pruning reduces computation cost by dropping neurons altogether, layer freezing keeps all its neurons and only performs forward passes only, thus cutting down the amount of needed computation in half. CoCo-FL is a framework that employs the concept of layer freezing proposed by Pfeiffer et al. [149]. Besides freezing layers, the authors suggested reducing computation further by quantizing frozen layers during forward and/or backward passes based on certain criteria. Whereas an arbitrary manual method of configuring local DNN models (in terms of layer freezing and quantization) is feasible, it is highly impractical, therefore, the study utilized a heuristic optimization algorithm to select appropriate configurations for each local model during training. In a comparison among CoCo-FL, FedAVG [8] and HeteroFL [148], the first scored the fastest convergence rate while retaining close accuracy to FedAVG when tested on the CIFAR dataset [65].

5. Challenges and future scope

Despite the relatively young age of FL technology, there has been notable progress made in improving the efficiency of communication and computation. Nonetheless, there remains plenty of room for improvement. This section discusses open issues and challenges that are either yet to be addressed or lack comprehensive study to be ready for practical implementation.

5.1. Suitability for Massive Scale Networks

Perhaps one of the most glaring issues with the literature reviewed in this paper is the limited scale of implementation, evidenced by the small numbers of clients prepared in empirical testing simulations. In many practical implementations, FL might need to accommodate a massive number of devices. To better imagine the scale of such networks one may take a look at two implementations of FL implemented in recent years; the Google keyboard (Gboard), with more than a billion installs [150], and Apple's Siri preinstalled on hundreds of millions of iPhone devices around the globe.

As noted in Section 2, AIoT architectures and applications are intuitive candidates for FL utilization, particularly for its rule in privacy preservation. It is predicted by Transforma Insights that the number of connected IoT devices will reach 30.5 billion active devices by 2030 [151]. In such implementations, where the number of clients spans to such a massive quantity, it is uncertain if any of the techniques reviewed in this paper would be capable of withstanding this massive volume.

The study conducted by Bonawitz et al. is one of the few instances of an FL-oriented system designed for large-scale applications [31]. It addressed several communication issues such as instability of connections and resource limitations. The work was used as a building block for the generalized system seen in [152]. However, beyond these pieces of research, little work was put into adopting efficiency frameworks to massive FL networks that make use of mobile crowdsensing. Perhaps future research work could concentrate on such a task, and demonstrate framework efficacy through experimentations that involve large numbers of client devices, simulated or otherwise.

5.2. The security trade-off

Various trade-offs were observed and discussed in the work reviewed above; nevertheless, a less obvious one is the security-efficiency trade-off. The principal motivation behind FL is the preservation of user data, which is threatened in many cases by the efficiency solutions suggested by the various research work conducted on the topic. This can be noted in multiple solutions to the problem of client selection, where critical information about client devices and connection states are shared across the network prior to selection. Such information in multi-access networks such as those carried through wifi is especially vulnerable to interception and interference. Similarly, the same issue is echoed in solutions for resource allocation and hyperparameter optimization that require a similar exchange of information as seen in [53,58,127,129]. In massive networks where efficiency solutions are necessary, such as AIoT networks, ensuring awareness of privacy constraints is crucial. Security and privacy-aware FL solutions to communication and/or computation efficiency could perhaps be an interesting direction for future studies to focus on.

5.3. Framework unification

The methods discussed in previous sections brought forth solutions to a multitude of efficiency problems, but it remains uncertain if several of these solutions could be incorporated together to form unified frameworks for whatever required purpose, and if that would result in unfavored tradeoffs either in regards to computation or communication. To illustrate this point further, one might take a look at [62,105], and [127], all of which utilized DNN to solve the issue of client selection, updates compression, and resource allocation, respectively. Having all of these DNN models operating in the same framework could result in a large additional computation cost, due to the resource-hungry nature of DNN.

Bui et al. offered an example of a unified framework for federated and continual learning targeted particularly at variational inference [153]. The work combines multiple algorithms from the scattered literary work on the topic into one straightforward and coherent framework with impressive capabilities. Such a framework is desperately needed in efficient FL, plagued with fragmented research. Additionally, future research could study the extent to which certain approaches could be coupled into encompassing frameworks.

5.4. Standarized benchmarking

Benchmarking in the context of machine learning refers to a set of standardized tools employed for the rigorous evaluation of models in comparison to other models of the same class. FL-orientated communication and computation research relies on a small variety of common datasets, which often do not properly convey the true performance of models in practice. For instance, most basic neural networks can show overwhelmingly high prediction accuracy with the MNIST [63], and CIFAR10 [65] datasets, which makes it hard to judge the true impact of the efficiency-performance trade-off. This is a special variability to FL since it relies on client-collected datasets, which are most likely not as tidy as most of the widely used testing datasets.

Furthermore, there is a grave lack of standardization of benchmarking in the field of FL in general and efficiency research in specific. It is hard to make out the extent to which a communication or computation framework might be more efficient than another when aspects of comparison almost always differ from one work to another. Therefore, the community is in dire need for a standard benchmark framework capable of effectively measuring the practical performance of FL frameworks.

6. Summary

Ever since its emergence in late 2015, Federated Learning has witnessed a fast-paced surge in global research interest. a significant chunk of said research was dedicated towards finding solutions to improve communication and/or computation efficiency which are very crucial in the practical implementation of the technology. This review paper provides a comprehensive, systematic overview of the current state of research on the topic, covering in-depth and breadth a wide variety of recent solutions and techniques. Particularly, we construct a convenient taxonomy that breaks down and encompasses the numerous techniques, frameworks, and algorithms developed to address the issue. Communication efficiency research was categorized into client selection, updates compression, updates dropping, network topology, and over-the-air aggregation; whereas computation efficiency research was divided into resource allocation, client selection, hyperparameter optimization, and model pruning. Lastly, we discuss the most pressing current challenges facing the field, based on which we propose several possible directions for future research work.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Chee-Onn Chow reports financial support was provided by Malaysia Ministry of Higher Education.

Data availability

No data was used for the research described in the article.

Acknowledgments

This project was supported by the Ministry of Higher Education (MOHE) Malaysia under the Fundamental Research Grant Scheme (FRGS) (Grant number: FRGS/1/2020/TK0/UM/02/4)

References

[1] D. Zhang, N. Maslej, E. Brynjolfsson, J. Etchemendy, T. Lyons, J. Manyika, H. Ngo, J.C. Niebles, M. Sellitto, E. Sakhaee, Y. Shoham, J. Clark, R. Perrault, The AI index 2022 annual report, 2022.
[2] S. Secinaro, D.M. Calandra, A. Secinaro, V. Muthurangu, P.P. Biancone, The role of artificial intelligence in healthcare: a structured literature review, BMC Med. Inform. Decis. Mak. 21 (2021).
[3] B. Vlačić, L. Corbo, S. Costa e Silva, M. Dabić, The evolving role of artificial intelligence in marketing: A review and research agenda, J. Bus. Res. 128 (2021) 187–203.
[4] H. Wu, H. Han, X. Wang, S. Sun, Research on artificial intelligence enhancing internet of things security: A survey, IEEE Access 8 (2020) 153826–153848.
[5] M. Merenda, C. Porcaro, D. Iero, Edge machine learning for AI-enabled IoT devices: A review, Sensors 20 (9) (2020).
[6] T. van der Ploeg, P.C. Austin, E.W. Steyerberg, Modern modelling techniques are data hungry: a simulation study for predicting dichotomous endpoints, BMC Med. Res. Methodol. 14 (2014).
[7] 2018 reform of EU data protection rules.
[8] H.B. McMahan, E. Moore, D. Ramage, B.A. y Arcas, Federated learning of deep networks using model averaging, 2016, CoRR arXiv:1602.05629, URL http://arxiv.org/abs/1602.05629.
[9] M. Asad, A. Moustafa, T. Ito, M. Aslam, Evaluating the communication efficiency in federated learning algorithms, in: 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design, CSCWD, 2021, pp. 552–557, http://dx.doi.org/10.1109/CSCWD49262.2021.9437738.
[10] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, M. Guizani, A survey on federated learning: The journey from centralized to distributed on-site learning and beyond, IEEE Internet Things J. 8 (7) (2021) 5476–5497.
[11] M. Aledhari, R. Razzak, R.M. Parizi, F. Saeed, Federated learning: A survey on enabling technologies, protocols, and applications, IEEE Access 8 (2020) 140699–140725.
[12] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.
[13] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, X. Liu, B. He, A survey on federated learning systems: Vision, hype and reality for data privacy and protection, 2019, CoRR arXiv:1907.09693, URL http://arxiv.org/abs/1907.09693.
[14] O. Shahid, S. Pouriyeh, R.M. Parizi, Q.Z. Sheng, G. Srivastava, L. Zhao, Communication efficiency in federated learning: Achievements and challenges, 2021, CoRR arXiv:2107.10996, URL https://arxiv.org/abs/2107.10996.
[15] L.U. Khan, W. Saad, Z. Han, E. Hossain, C.S. Hong, Federated learning for internet of things: Recent advances, taxonomy, and open challenges, 2020, CoRR arXiv:2009.13012, URL https://arxiv.org/abs/2009.13012.
[16] S.R. Kurupathi, W. Maass, Survey on federated learning towards privacy preserving AI, in: Computer Science: Information Technology, AIRCC Publishing Corporation, 2020.
[17] D.H. Mahlool, M.H. Abed, A comprehensive survey on federated learning: Concept and applications, 2022, CoRR arXiv:2201.09384, URL https://arxiv.org/abs/2201.09384.
[18] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for internet of things: A comprehensive survey, IEEE Commun. Surv.: Tutorials 23 (3) (2021) 1622–1658.
[19] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 2031–2063.
[20] J. Konečný, H.B. McMahan, D. Ramage, P. Richtárik, Federated optimization: Distributed machine learning for on-device intelligence, 2016, CoRR arXiv:1610.02527, URL http://arxiv.org/abs/1610.02527.
[21] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2015, pp. 909–910, http://dx.doi.org/10.1109/ALLERTON.2015.7447103.
[22] Q. Li, Z. Wen, B. He, Practical federated gradient boosting decision trees, 2019, CoRR arXiv:1911.04206, URL http://arxiv.org/abs/1911.04206.
[23] J. Kang, Z. Xiong, C. Jiang, Y. Liu, S. Guo, Y. Zhang, D. Niyato, C. Leung, C. Miao, Scalable and communication-efficient decentralized federated edge learning with multi-blockchain framework, 2020, CoRR arXiv:2008.04743, URL https://arxiv.org/abs/2008.04743.

[24] U. Majeed, C.S. Hong, Flchain: Federated learning via MEC-enabled blockchain network, in: 2019 20th Asia-Pacific Network Operations and Management Symposium, APNOMS, 2019, pp. 1–4, http://dx.doi.org/10.23919/APNOMS.2019.8892848.

[25] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, W. Luo, DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive, IEEE Trans. Dependable Secure Comput. 18 (5) (2021) 2438–2455.

[26] L. Cui, X. Su, Z. Ming, Z. Chen, S. Yang, Y. Zhou, W. Xiao, CREAT: Blockchain-assisted compression algorithm of federated learning for content caching in edge computing, IEEE Internet Things J. (2020) 1.

[27] Y. Sun, J. Shao, Y. Mao, J. Zhang, Semi-decentralized federated edge learning for fast convergence on non-IID data, 2021, CoRR arXiv:2104.12678, URL https://arxiv.org/abs/2104.12678.

[28] K. Nandury, A. Mohan, F. Weber, Cross-silo federated training in the cloud with diversity scaling and semi-supervised learning, in: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2021, pp. 3085–3089, http://dx.doi.org/10.1109/ICASSP39728.2021.9413428.

[29] G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J.V. Mancuso, F. Jungmann, M.-M. Steinborn, A. Saleh, M.R. Makowski, D. Rueckert, R.F. Braren, End-to-end privacy preserving deep learning on multi-institutional medical imaging, Nat. Mach. Intell. 3 (2021) 473–484.

[30] A. Durrant, M. Markovic, D. Matthews, D. May, J.A. Enright, G. Leontidis, The role of cross-silo federated learning in facilitating data sharing in the agri-food sector, 2021, CoRR arXiv:2104.07468, URL https://arxiv.org/abs/2104.07468.

[31] K.A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H.B. McMahan, T.V. Overveldt, D. Petrou, D. Ramage, J. Roselander, Towards federated learning at scale: System design, 2019, CoRR arXiv:1902.01046, URL http://arxiv.org/abs/1902.01046.

[32] W.Y.B. Lim, J. Huang, Z. Xiong, J. Kang, D.T. Niyato, X. Hua, C. Leung, C. Miao, Towards federated learning in UAV-enabled internet of vehicles: A multi-dimensional contract-matching approach, IEEE Trans. Intell. Transp. Syst. 22 (2021) 5140–5154.

[33] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, 2019, CoRR arXiv:1902.04885, URL http://arxiv.org/abs/1902.04885.

[34] K. Wei, J. Li, C. Ma, M. Ding, S. Wei, F. Wu, G. Chen, T. Ranbaduge, Vertical federated learning: Challenges, methodologies and experiments, 2022, CoRR arXiv:2202.04309, URL https://arxiv.org/abs/2202.04309.

[35] A.A. Abdellatif, N. Mhaisen, A. Mohamed, A. Erbad, M. Guizani, Z. Dawy, W. Nasreddine, Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data, 2021, CoRR arXiv:2107.06548, URL https://arxiv.org/abs/2107.06548.

[36] L. Liu, J. Zhang, S. Song, K.B. Letaief, Client-edge-cloud hierarchical federated learning, in: ICC 2020 - 2020 IEEE International Conference on Communications, ICC, 2020, pp. 1–6, http://dx.doi.org/10.1109/ICC40277.2020.9148862.

[37] J. Tursunboev, Y.-S. Kang, S.-B. Huh, D.-W. Lim, J.-M. Kang, H. Jung, Hierarchical federated learning for edge-aided unmanned aerial vehicle networks, Appl. Sci. 12 (2) (2022).

[38] M.S.H. Abad, E. Ozfatura, D. GUndUz, O. Ercetin, Hierarchical federated learning ACROSS heterogeneous cellular networks, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2020, pp. 8866–8870, http://dx.doi.org/10.1109/ICASSP40776.2020.9054634.

[39] L. Hannah, Stochastic optimization, Int. Encycl. Soc. Behav. Sci. 2 (2015).

[40] W. Chen, S. Horváth, P. Richtárik, Optimal client sampling for federated learning, 2020, CoRR arXiv:2010.13723, URL https://arxiv.org/abs/2010.13723.

[41] Y.J. Cho, J. Wang, G. Joshi, Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, 2020, CoRR, arXiv:2010.01243, URL https://arxiv.org/abs/2010.01243.

[42] M. Mitzenmacher, The power of two choices in randomized load balancing, IEEE Trans. Parallel Distrib. Syst. 12 (2001) 1094–1104.

[43] H.T. Nguyen, V. Sehwag, S. Hosseinalipour, C.G. Brinton, M. Chiang, H.V. Poor, Fast-convergent federated learning, 2020, CoRR arXiv:2007.13137, URL https://arxiv.org/abs/2007.13137.

[44] A.K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, V. Smith, On the convergence of federated optimization in heterogeneous networks, 2018, CoRR arXiv:1812.06127, URL http://arxiv.org/abs/1812.06127.

[45] Z. Chen, K.F.E. Chong, T.Q.S. Quek, Dynamic attention-based communication-efficient federated learning, 2021, CoRR arXiv:2108.05765, URL https://arxiv.org/abs/2108.05765.

[46] T. Huang, W. Lin, K. Li, A.Y. Zomaya, Stochastic client selection for federated learning with volatile clients, 2020, CoRR arXiv:2011.08756, URL https://arxiv.org/abs/2011.08756.

[47] P. Auer, N. Cesa-Bianchi, Y. Freund, R. Schapire, Gambling in a rigged casino: The adversarial multi-armed bandit problem, in: Proceedings of IEEE 36th Annual Foundations of Computer Science, 1995, pp. 322–331, http://dx.doi.org/10.1109/SFCS.1995.492488.

[48] J. Xu, H. Wang, Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective, 2020, CoRR arXiv:2004.04314, URL https://arxiv.org/abs/2004.04314.

[49] S.A. Curtis, The classification of greedy algorithms, Sci. Comput. Progr. 49 (1) (2003) 125–157.

[50] R. Balakrishnan, T. Li, T. Zhou, N. Himayat, V. Smith, J. Bilmes, Diverse client selection for federated learning via submodular maximization, in: International Conference on Learning Representations, 2022, URL https://openreview.net/forum?id=nwKXyFvaUm.

[51] G. Cornuejols, M. Fisher, G.L. Nemhauser, On the uncapacitated location problem**this research was supported by NSF grants ENG75-00568 and SOC-7402516. Sections 1–4 of this paper include a technical summary of some results given in [2]. Some proofs are omitted and may be obtained in [2], in: P. Hammer, E. Johnson, B. Korte, G. Nemhauser (Eds.), Studies in Integer Programming, in: Annals of Discrete Mathematics, Vol. 1, Elsevier, 1977, pp. 163–177.

[52] S. Caldas, P. Wu, T. Li, J. Konečný, H.B. McMahan, V. Smith, A. Talwalkar, LEAF: a benchmark for federated settings, 2018, CoRR arXiv:1812.01097, URL http://arxiv.org/abs/1812.01097.

[53] S. Abdulrahman, H. Tout, A. Mourad, C. Talhi, Fedmccs: Multicriteria client selection model for optimal IoT federated learning, IEEE Internet Things J. 8 (6) (2021) 4723–4735.

[54] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019 - 2019 IEEE International Conference on Communications, ICC, 2019, pp. 1–7, http://dx.doi.org/10.1109/ICC.2019.8761315.

[55] G. Mohi-ud din, NSL-KDD, 2018.

[56] L. Wang, W. Wang, B. Li, CMFL: Mitigating communication overhead for federated learning, in: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, 2019, pp. 954–964, http://dx.doi.org/10.1109/ICDCS.2019.00099.

[57] P. Klemperer, Auction theory: A guide to the literature, J. Econ. Surv. 13 (3) (1999) 227–286.

[58] T.H. Thi Le, N.H. Tran, Y.K. Tun, M.N.H. Nguyen, S.R. Pandey, Z. Han, C.S. Hong, An incentive mechanism for federated learning in wireless cellular networks: An auction approach, IEEE Trans. Wireless Commun. 20 (8) (2021) 4874–4887.

[59] J. Zhang, Y. Wu, R. Pan, Incentive mechanism for horizontal federated learning based on reputation and reverse auction, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 947–956.

[60] J. Zhang, Y. Wu, R. Pan, Auction-based ex-post-payment incentive mechanism design for horizontal federated learning with reputation and contribution measurement, 2022, CoRR arXiv:2201.02410, URL https://arxiv.org/abs/2201.02410.

[61] J. Zhang, Y. Wu, R. Pan, Online auction-based incentive mechanism design for horizontal federated learning with budget constraint, 2022, CoRR arXiv:2201.09047, URL https://arxiv.org/abs/2201.09047.

[62] P. Zhang, C. Wang, C. Jiang, Z. Han, Deep reinforcement learning assisted federated learning algorithm for data management of IIoT, IEEE Trans. Ind. Inform. 17 (12) (2021) 8475–8484.
[63] Y. LeCun, C. Cortes, MNIST handwritten digit database, 2010.
[64] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017, URL http://arxiv.org/abs/1708.07747, cite arXiv:1708.07747 Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/.
[65] A. Krizhevsky, V. Nair, G. Hinton, CIFAR-10 (Canadian Institute for Advanced Research).
[66] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-IID data with reinforcement learning, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 1698–1707, http://dx.doi.org/10.1109/INFOCOM41043.2020.9155494.
[67] S.Q. Zhang, J. Lin, Q. Zhang, A multi-agent reinforcement learning approach for efficient client selection in federated learning, 2022, CoRR arXiv:2201.02932, URL https://arxiv.org/abs/2201.02932.
[68] S. Park, Y. Suh, J. Lee, Fedpso: Federated learning using particle swarm optimization to reduce communication costs, Sensors 21 (2) (2021).
[69] S. Caldas, J. Konečný, H.B. McMahan, A. Talwalkar, Expanding the reach of federated learning by reducing client resource requirements, 2018, CoRR arXiv:1812.07210, URL http://arxiv.org/abs/1812.07210.
[70] A. Malekijoo, M.J. Fadaeieslam, H. Malekijou, M. Homayounfar, F. Alizadeh-Shabdiz, R. Rawassizadeh, FEDZIP: a compression framework for communication-efficient federated learning, 2021, CoRR arXiv:2102.01593, URL https://arxiv.org/abs/2102.01593.
[71] A.F. Aji, K. Heafield, Sparse communication for distributed gradient descent, 2017, CoRR arXiv:1704.05021, URL http://arxiv.org/abs/1704.05021.
[72] D.A. Huffman, A method for the construction of minimum-redundancy codes, Resonance 11 (1952) 91–99.
[73] N. Shlezinger, M. Chen, Y.C. Eldar, H.V. Poor, S. Cui, UVeQFed: Universal vector quantization for federated learning, IEEE Trans. Signal Process. 69 (2021) 500–514.
[74] F. Sattler, S. Wiedemann, K. Müller, W. Samek, Robust and communication-efficient federated learning from non-iid data, 2019, CoRR arXiv:1903.02891, URL http://arxiv.org/abs/1903.02891.
[75] F. Sattler, S. Wiedemann, K. Müller, W. Samek, Sparse binary compression: Towards distributed deep learning with minimal communication, 2018, CoRR arXiv:1805.08768, URL http://arxiv.org/abs/1805.08768.
[76] S. Golomb, Run-length encodings (corresp.), IEEE Trans. Inform. Theory 12 (3) (1966) 399–401.
[77] J. Xu, W. Du, R. Cheng, W. He, Y. Jin, Ternary compression for communication-efficient federated learning, 2020, CoRR arXiv:2003.03564, URL https://arxiv.org/abs/2003.03564.
[78] C. Zhu, S. Han, H. Mao, W.J. Dally, Trained ternary quantization, 2016, CoRR arXiv:1612.01064, URL http://arxiv.org/abs/1612.01064.
[79] S. Zheng, C. Shen, X. Chen, Design and analysis of uplink and downlink communications for federated learning, 2020, CoRR arXiv:2012.04057, URL https://arxiv.org/abs/2012.04057.
[80] N. Shlezinger, M. Chen, Y.C. Eldar, H.V. Poor, S. Cui, Federated learning with quantization constraints, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2020, pp. 8851–8855, http://dx.doi.org/10.1109/ICASSP40776.2020.9054168.
[81] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, R. Pedarsani, Robust and communication-efficient collaborative learning, 2019, CoRR arXiv:1907.10595, URL http://arxiv.org/abs/1907.10595.
[82] H. Tang, S. Gan, C. Zhang, T. Zhang, J. Liu, Communication compression for decentralized training, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.
[83] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, R. Pedarsani, Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization, 2019, CoRR arXiv:1909.13014, URL http://arxiv.org/abs/1909.13014.
[84] L. Liu, J. Zhang, S. Song, K.B. Letaief, Hierarchical quantized federated learning: Convergence analysis and system design, 2021, CoRR arXiv:2103.14272, URL https://arxiv.org/abs/2103.14272.
[85] M.M. Amiri, D. Gündüz, S.R. Kulkarni, H.V. Poor, Federated learning with quantized global model updates, 2020, CoRR arXiv:2006.10672, URL https://arxiv.org/abs/2006.10672.
[86] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, R. Arora, Fetchsgd: Communication-efficient federated learning with sketching, 2020, CoRR arXiv:2007.07682, URL https://arxiv.org/abs/2007.07682.
[87] M. Charikar, K. Chen, M. Farach-Colton, Finding frequent items in data streams, in: Proceedings of the 29th International Colloquium on Automata, Languages and Programming, ICALP '02, Springer-Verlag, Berlin, Heidelberg, 2002, pp. 693–703.
[88] N. Ivkin, D. Rothchild, E. Ullah, V. braverman, I. Stoica, R. Arora, Communication-efficient distributed SGD with sketching, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.
[89] Z. Liu, T. Li, V. Smith, V. Sekar, Enhancing the privacy of federated learning with sketching, 2019, CoRR arXiv:1911.01812, URL http://arxiv.org/abs/1911.01812.
[90] J. Zhao, H. Zhu, F. Wang, R. Lu, H. Li, J. Tu, J. Shen, CORK: A privacy-preserving and lossless federated learning scheme for deep neural network, Inform. Sci. 603 (2022) 190–209.
[91] Y. He, M. Zenk, M. Fritz, Cossgd: Nonlinear quantization for communication-efficient federated learning, 2020, CoRR arXiv:2012.08241, URL https://arxiv.org/abs/2012.08241.
[92] M. Rani, S.B. Dhok, R.B. Deshmukh, A systematic review of compressive sensing: Concepts, implementations and applications, IEEE Access 6 (2018) 4875–4894.
[93] Y.-S. Jeon, M.M. Amiri, J. Li, H.V. Poor, A compressive sensing approach for federated learning over massive mimo communication systems, IEEE Trans. Wireless Commun. 20 (3) (2021) 1990–2004.
[94] C. Li, G. Li, P.K. Varshney, Communication-efficient federated learning based on compressed sensing, IEEE Internet Things J. 8 (20) (2021) 15531–15541.
[95] T. Blumensath, M.E. Davies, Iterative hard thresholding for compressed sensing, Appl. Comput. Harmon. Anal. 27 (3) (2009) 265–274.
[96] J. Bernstein, Y. Wang, K. Azizzadenesheli, A. Anandkumar, Signsgd: compressed optimisation for non-convex problems, 2018, CoRR arXiv:1802.04434, URL http://arxiv.org/abs/1802.04434.
[97] Y. Oh, N. Lee, Y.-S. Jeon, Quantized compressed sensing for communication-efficient federated learning, in: 2021 IEEE Globecom Workshops (GC Wkshps), 2021, pp. 1–6, http://dx.doi.org/10.1109/GCWkshps52748.2021.9682076.
[98] J.P. Vila, P. Schniter, Expectation-maximization Gaussian-mixture approximate message passing, IEEE Trans. Signal Process. 61 (19) (2013) 4658–4672.
[99] X. Fan, Y. Wang, Y. Huo, Z. Tian, 1-bit compressive sensing for efficient federated learning over the air, 2021, CoRR arXiv:2103.16055, URL https://arxiv.org/abs/2103.16055.
[100] A. Abdi, F. Fekri, Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, (04) 2020, pp. 3105–3112, http://dx.doi.org/10.1609/aaai.v34i04.5706.
[101] Y. Oh, Y.-S. Jeon, M. Chen, W. Saad, Fedvqcs: Federated learning via vector quantized compressed sensing, 2022.
[102] N. Bouacida, J. Hou, H. Zang, X. Liu, Adaptive federated dropout: Improving communication efficiency and generalization for federated learning, 2020, CoRR arXiv:2011.04050, URL https://arxiv.org/abs/2011.04050.
[103] H. Zhu, Y. Jin, Multi-objective evolutionary federated learning, 2018, CoRR arXiv:1812.07478, URL http://arxiv.org/abs/1812.07478.
[104] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[105] C. Wu, F. Wu, R. Liu, L. Lyu, Y. Huang, X. Xie, Fedkd: Communication efficient federated learning via knowledge distillation, 2021, CoRR arXiv:2108.13323, URL https://arxiv.org/abs/2108.13323.

[106] Y. Chen, X. Sun, Y. Jin, Communication-efficient federated deep learning with asynchronous model update and temporally weighted aggregation, 2019, CoRR arXiv:1903.07424, URL http://arxiv.org/abs/1903.07424.

[107] C. Hu, J. Jiang, Z. Wang, Decentralized federated learning: A segmented gossip approach, 2019, CoRR arXiv:1908.07782, URL http://arxiv.org/abs/1908.07782.

[108] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, D. Lewin, Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web, in: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC '97, Association for Computing Machinery, New York, NY, USA, 1997, pp. 654–663.

[109] Z. Wang, Y. Hu, S. Yan, Z. Wang, R. Hou, C. Wu, Efficient ring-topology decentralized federated learning with deep generative models for medical data in ehealthcare systems, Electronics 11 (10) (2022).

[110] A. Bellet, A. Kermarrec, E. Lavoie, D-cliques: Compensating NonIIDness in decentralized federated learning with topology, 2021, CoRR arXiv:2104.07365, URL https://arxiv.org/abs/2104.07365.

[111] C. Giusti, E. Pastalkova, C. Curto, V. Itskov, Clique topology reveals intrinsic geometric structure in neural correlations, Proc. Natl. Acad. Sci. 112 (44) (2015) 13455–13460.

[112] H. Yang, H-FL: a hierarchical communication-efficient and privacy-protected architecture for federated learning, 2021, CoRR arXiv:2106.00275, URL https://arxiv.org/abs/2106.00275.

[113] L. Su, V.K.N. Lau, Hierarchical federated learning for hybrid data partitioning across multitype sensors, IEEE Internet Things J. 8 (13) (2021) 10922–10939.

[114] O. Marfoq, C. Xu, G. Neglia, R. Vidal, Throughput-optimal topology design for cross-silo federated learning, 2020, CoRR arXiv:2010.12229, URL https://arxiv.org/abs/2010.12229.

[115] H. Gao, H. Huang, Periodic stochastic gradient descent with momentum for decentralized training, 2020, CoRR arXiv:2008.10435, URL https://arxiv.org/abs/2008.10435.

[116] J. Wang, A.K. Sahu, Z. Yang, G. Joshi, S. Kar, MATCHA: speeding up decentralized SGD via matching decomposition sampling, 2019, CoRR arXiv:1905.09435, URL http://arxiv.org/abs/1905.09435.

[117] K. Yang, T. Jiang, Y. Shi, Z. Ding, Federated learning via over-the-air computation, IEEE Trans. Wireless Commun. 19 (3) (2020) 2022–2035.

[118] G. Zhu, Y. Du, D. Gündüz, K. Huang, One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis, IEEE Trans. Wireless Commun. 20 (3) (2021) 2120–2135.

[119] T. Sery, N. Shlezinger, K. Cohen, Y.C. Eldar, Over-the-air federated learning from heterogeneous data, IEEE Trans. Signal Process. 69 (2021) 3796–3811.

[120] S. Wang, Y. Hong, R. Wang, Q. Hao, Y.-C. Wu, D.W.K. Ng, Edge federated learning via unit-modulus over-the-air computation, IEEE Trans. Commun. 70 (5) (2022) 3141–3156.

[121] H. Hellström, V. Fodor, C. Fischione, Over-the-air federated learning with retransmissions, in: 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications, SPAWC, 2021, pp. 291–295, http://dx.doi.org/10.1109/SPAWC51858.2021.9593119.

[122] K. Yang, T. Jiang, Y. Shi, Z. Ding, Federated learning based on over-the-air computation, in: ICC 2019 - 2019 IEEE International Conference on Communications, ICC, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICC.2019.8761429.

[123] D.E. Knuth, Big omicron and big omega and big theta, ACM Sigact News 8 (2) (1976) 18–24.

[124] T.D. Burd, R.W. Brodersen, Processor design for portable systems, J. VLSI Signal Process. Syst. Signal, Image Video Technol. 13 (1996) 203–221.

[125] N.H. Tran, W. Bao, A. Zomaya, M.N.H. Nguyen, C.S. Hong, Federated learning over wireless networks: Optimization model design and analysis, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1387–1395, http://dx.doi.org/10.1109/INFOCOM.2019.8737464.

[126] Z. Yang, M. Chen, W. Saad, C.S. Hong, M. Shikh-Bahaei, Energy efficient federated learning over wireless communication networks, IEEE Trans. Wireless Commun. 20 (3) (2021) 1935–1949.

[127] T.T. Anh, N.C. Luong, D. Niyato, D.I. Kim, L. Wang, Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach, 2018, CoRR arXiv:1812.03633, URL http://arxiv.org/abs/1812.03633.

[128] H.T. Nguyen, N.C. Luong, J. Zhao, C. Yuen, D. Niyato, Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach, 2019, CoRR arXiv:1910.09172, URL http://arxiv.org/abs/1910.09172.

[129] Y. Sarikaya, Ö. Erçetin, Motivating workers in federated learning: A stackelberg game perspective, 2019, CoRR arXiv:1908.03092, URL http://arxiv.org/abs/1908.03092.

[130] S.R. Pandey, N.H. Tran, M. Bennis, Y.K. Tun, A. Manzoor, C.S. Hong, A crowdsourcing framework for on-device federated learning, IEEE Trans. Wireless Commun. 19 (5) (2020) 3241–3256.

[131] R. Hu, Y. Gong, Trading data for learning: Incentive mechanism for on-device federated learning, 2020, CoRR arXiv:2009.05604, URL https://arxiv.org/abs/2009.05604.

[132] G. Xiao, M. Xiao, G. Gao, S. Zhang, H. Zhao, X. Zou, Incentive mechanism design for federated learning: A two-stage stackelberg game approach, in: 2020 IEEE 26th International Conference on Parallel and Distributed Systems, ICPADS, 2020, pp. 148–155, http://dx.doi.org/10.1109/ICPADS51040.2020.00029.

[133] Y. Zhu, Z. Liu, P. Wang, C. Du, A dynamic incentive and reputation mechanism for energy-efficient federated learning in 6g, Digit. Commun. Netw. (2022) URL https://www.sciencedirect.com/science/article/pii/S2352864822000542.

[134] A. Taïk, H. Moudoud, S. Cherkaoui, Data-quality based scheduling for federated edge learning, in: 2021 IEEE 46th Conference on Local Computer Networks, LCN, 2021, pp. 17–23, http://dx.doi.org/10.1109/LCN52139.2021.9524974.

[135] S. Rai, A. Kumari, D.K. Prasad, Client selection in federated learning under imperfections in environment, AI 3 (1) (2022) 124–145.

[136] L. Nagalapatti, R.S. Mittal, R. Narayanam, Is your data relevant?: Dynamic selection of relevant data for federated learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, (7) 2022, pp. 7859–7867, http://dx.doi.org/10.1609/aaai.v36i7.20755.

[137] D. Preuveneers, I. Tsingenopoulos, W. Joosen, Resource usage and performance trade-offs for machine learning models in smart environments, Sensors 20 (4) (2020) 1176.

[138] B. Qolomany, K. Ahmad, A.I. Al-Fuqaha, J. Qadir, Particle swarm optimized federated learning for industrial IoT and smart city services, 2020, CoRR arXiv:2009.02560, URL https://arxiv.org/abs/2009.02560.

[139] Y. Zhou, P. Ram, T. Salonidis, N. Baracaldo, H. Samulowitz, H. Ludwig, Single-shot hyper-parameter optimization for federated learning: A general algorithm analysis, 2022.

[140] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[141] S. Agrawal, S. Sarkar, M. Alazab, P.K.R. Maddikunta, T.R. Gadekallu, Q. Pham, Genetic CFL: optimization of hyper-parameters in clustered federated learning, 2021, CoRR arXiv:2107.07233, URL https://arxiv.org/abs/2107.07233.

[142] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, AAAI Press, 1996, pp. 226–231.

[143] Y. Zeng, X. Wang, J. Yuan, J. Zhang, J. Wan, J. Xiong, Local epochs inefficiency caused by device heterogeneity in federated learning, Wirel. Commun. Mob. Comput. 2022 (2022).

[144] J. Wang, G. Joshi, Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD, 2018, CoRR arXiv:1810.08313, URL http://arxiv.org/abs/1810.08313.
[145] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015, CoRR arXiv:1512.03385, URL http://arxiv.org/abs/1512.03385.
[146] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv e-prints arXiv:1409.1556.
[147] Y. Jiang, S. Wang, V. Valls, B.J. Ko, W.-H. Lee, K.K. Leung, L. Tassiulas, Model pruning enables efficient federated learning on edge devices, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–13.
[148] E. Diao, J. Ding, V. Tarokh, Heterofl: Computation and communication efficient federated learning for heterogeneous clients, 2020, CoRR arXiv:2010.01264, URL https://arxiv.org/abs/2010.01264.
[149] K. Pfeiffer, M. Rapp, R. Khalili, J. Henkel, Cocofl: Communication- and computation-aware federated learning via partial NN freezing and quantization, 2022.
[150] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, D. Ramage, Federated learning for mobile keyboard prediction, 2018, CoRR arXiv:1811.03604, URL http://arxiv.org/abs/1811.03604.
[151] Transforma Insights, Current IOT forecast highlights, 2020, Online; (accessed 25 January 2023).
[152] M. Paulik, M. Seigel, H. Mason, D. Telaar, J. Kluivers, R.C. van Dalen, C.W. Lau, L. Carlson, F. Granqvist, C. Vandevelde, S. Agarwal, J. Freudiger, A. Byde, A. Bhowmick, G. Kapoor, S. Beaumont, Á. Cahill, D. Hughes, O. Javidbakht, F. Dong, R. Rishi, S. Hung, Federated evaluation and tuning for on-device personalization: System design & applications, 2021, CoRR arXiv:2102.08503, URL https://arxiv.org/abs/2102.08503.
[153] T.D. Bui, C.V. Nguyen, S. Swaroop, R.E. Turner, Partitioned variational inference: A unified framework encompassing federated and continual learning, 2018, arXiv:1811.11206.