Communication-Efficient Edge AI: Algorithms and Systems

Yuanming Shi, *Member, IEEE*, Kai Yang, *Graduate Student Member, IEEE*, Tao Jiang, *Graduate Student Member, IEEE*, Jun Zhang, *Senior Member, IEEE*, and Khaled B. Letaief, *Fellow, IEEE*

***Abstract*—Artificial intelligence (AI) has achieved remarkable breakthroughs in a wide range of fields, ranging from speech processing, image classification to drug discovery. This is driven by the explosive growth of data, advances in machine learning (especially deep learning), and the easy access to powerful computing resources. Particularly, the wide scale deployment of edge devices (e.g., IoT devices) generates an unprecedented scale of data, which provides the opportunity to derive accurate models and develop various intelligent applications at the network edge. However, such enormous data cannot all be sent to the cloud for processing, due to the varying channel quality, traffic congestion and/or privacy concerns, and the enormous energy consumption. By pushing inference and training processes of AI models to edge nodes, edge AI has emerged as a promising alternative. AI at the edge requires close cooperation among *edge devices*, such as smart phones and smart vehicles, and *edge servers* at the wireless access points and base stations, which however result in heavy communication overheads. In this paper, we present a comprehensive survey of the recent developments in various techniques for overcoming these communication challenges. Specifically, we first identify key communication challenges in edge AI systems. We then introduce communication-efficient techniques, from both algorithmic and system perspectives for training and inference tasks at the network edge. Potential future research directions are also highlighted.**

***Index Terms*—Artificial intelligence, edge AI, edge intelligence, communication efficiency.**

Manuscript received February 22, 2020; revised May 27, 2020; accepted July 1, 2020. Date of publication July 7, 2020; date of current version November 20, 2020. This work was supported by the National Nature Science Foundation of China under Grant 61601290. *(Corresponding author: Jun Zhang.)*

Yuanming Shi and Tao Jiang are with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: shiym@shanghaitech.edu.cn; jiangtao1@shanghaitech.edu.cn).

Kai Yang is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China, also with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: yangkai@shanghaitech.edu.cn).

Jun Zhang is with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong (e-mail: jun-eie.zhang@polyu.edu.hk).

Khaled B. Letaief is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, and also with Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: eekhaled@ust.hk).

Digital Object Identifier 10.1109/COMST.2020.3007787

I. Introduction

With the explosive growth in data and the rapid advancements of algorithms (e.g., deep learning), as well as the step-change improvement of computing resources, artificial intelligence (AI) has achieved breakthroughs in a wide range of applications, including speech processing [1], image classification [2] and reinforcement learning [3], etc. AI is expected to affect significant segments of many vertical industries and our daily life, such as intelligent vehicles [4] and tactile robots [5]. In addition, it is anticipated that AI could add around 16 percent or about $13 trillion to the global gross domestic product (GDP) by 2030, compared with that of 2018 [6].

The explosive data growth generated by the massive number of end devices, e.g., smart phones, tablets and Internet-of-Things (IoT) sensors, provides opportunities and challenges for providing intelligent services. It is predicted that there will be nearly 85 Zettabytes of usable data generated by all people, machines and things by 2021, which shall exceed the cloud data center traffic (21 Zettabytes) by a factor of 4 [7]. Moreover, delay-sensitive intelligent applications, such as autonomous driving, cyber-physical control systems, and robotics, require fast processing of the incoming data. Such extremely high network bandwidth and low latency requirements would place unprecedented pressures on traditional cloud-based AI, where massive sensors/embedded devices transfer collected data to the cloud [8], often under varying network qualities (e.g., the bandwidth and latency). In addition, privacy is a major concern for cloud-based solutions. To address these problems, one promising solution, edge AI [9], [10], comes to the rescue.

Futuristic wireless systems [11] mainly consist of ultra-dense edge nodes, including *edge servers* at the base stations and wireless access points, and *edge devices* such as smart phones, smart vehicles, and drones. Edge AI pushes inference and training processes of AI models to the network edge in close proximity to data sources. As such, the amount of data transferred to the cloud will be significantly reduced, thus alleviating the network traffic load, latency and privacy concerns. Although training an AI model (e.g., deep neural networks) generally requires intensive computing resources, the rapid development of mobile edge computing can provide cloud-computing capabilities at the edge of the mobile network [12], [13], making the application of AI to edges much more efficient [14]. In addition, computational

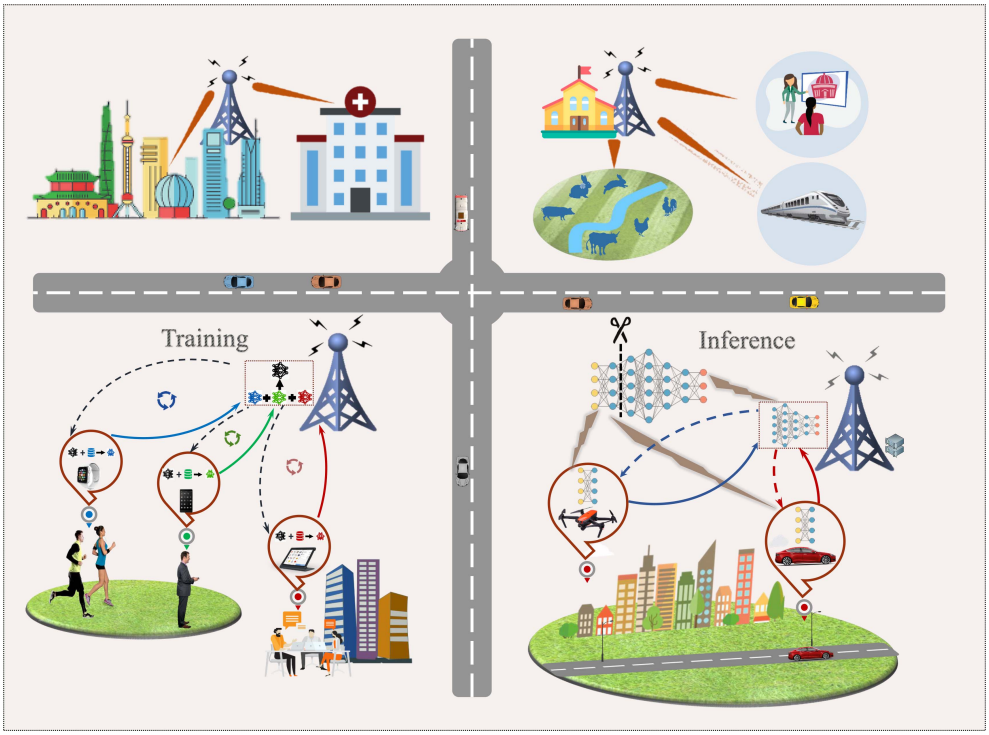


Fig. 1. Illustration of edge AI, including edge training and edge inference.

capabilities of edge servers and edge devices continue to improve. Notable examples include the deployment of neural processing unit (NPU) in Kirin 970 smart phone chips and the Apple's bionic chip A12, which substantially accelerate AI computations on edge devices. In a nutshell, the advances of mobile edge computing platforms and the improvement of computing power of the edge nodes make edge AI a feasible solution.

Nevertheless, pushing AI towards the edge is a non-trivial task. The most straightforward way of realizing edge AI without any communication load, i.e., deploying the full AI models on edge devices, is often infeasible when the size of the AI model (e.g., deep neural networks) is too large or the computational requirement is too high, given the limited hardware resources of edge devices. A promising solution is to incorporate cooperation among edge nodes to accomplish edge AI tasks that require intensive computation and large storage sizes. This can be achieved by exploiting different data storage and processing capabilities for a wider range of intelligent services with distinct latency and bandwidth requirements [15], as shown in Fig. 1. For example, based on federated learning [16], we can use multiple devices to train an AI model collaboratively. Specifically, each device only needs to compute a local model according to its own data samples, before sending the computation results to a fusion center, where the global AI model is aggregated and updated. The new AI model will be transmitted back to each device for training at the next epoch. Such solutions exploit on-device computing power in a collaborative way, which, however, requires significant communication overheads during the model updating process. In addition, some computation-intensive AI inference tasks can only be accomplished by task-splitting among edge devices and edge servers [17], which also incurs heavy communication cost. Therefore, the enormous communication overhead presents a major bottleneck for edge AI.

To unleash its full potential, the upcoming edge AI [9], [18] shall rely on advances in various aspects, including the smart design of distributed learning algorithms and system architectures, supported by efficient communication protocols. In this article, we survey the communication challenges in designing and deploying AI models and algorithms in edge AI systems. Specifically, we provide a thorough survey on communication-efficient distributed learning algorithms for training AI models on edges. In addition, we provide an overview of edge AI system architectures for communication-efficient edge training and edge inference. In the next section, we start with the motivations and identify major communication challenges in edge AI. We have summarized the acronyms across the paper in Table I. A paper outline is provided in Table II.

II. Motivations and Challenges

In this section, we present the motivations and identify key communication challenges of edge AI. Interplays among computation mechanisms, learning algorithms, as well as system architectures, are revealed.

A. Motivations

During the past decades, the thriving mobile Internet has enabled various mobile applications such as mobile pay, mobile gaming, etc. These applications in turn led to an upsurge of mobile devices and mobile data, which prompts the prosperity of AI for greatly facilitating daily life. As the

TABLE I
LIST OF ACRONYMS USED IN THE PAPER

AI	artificial intelligence	LDGM	low density generator matrix
BS	base station	LDPC	low density parity-check
CNN	convolutional neural network	MDS	maximum distance separable
DC	difference-of-convex-functions	mMTC	massive machine type communications
DNN	deep neural networks	non-IID	not independently and identically distributed
eMBB	enhanced mobile broadband	NPU	neural processing unit
FedAvg	federated averaging	QSGD	quantized stochastic gradient descent
GDP	gross domestic product	RNN	recurrent neural network
GDPR	general data protection regulation	SGD	stochastic gradient descent
IoT	Internet-of-Things	SVM	support vector machine
IRS	intelligent reflecting surface	TPU	tensor processing unit
LAG	lazily aggregated gradient	URLLC	ultra-reliable low-latency communications
L-BFGS	limited-memory Broyden Fletcher Goldfarb Shanno		

key design target, the 5G roll-out has focused on several key services for *connected things*: enhanced mobile broadband (eMBB), ultra reliable low latency communications (URLLC), and massive machine type communications (mMTC). In contrast, futuristic 6G networks [11] will undergo a paradigm shift from *connected things* to *connected intelligence*. The network infrastructure of 6G is envisioned to fully exploit the potential of massive distributed devices and the data generated at network edges for supporting intelligent applications [9].

In recent years, a new trend is to move computation tasks from the cloud center towards network edges due to the increasing computing power of edge nodes [13]. In the upcoming 5G wireless systems, there are a growing number of edge nodes, varying from base stations (BSs) to various edge devices such as mobile phones, tablets, and IoT devices. The computational capabilities of edge mobile devices have seen substantial improvements thanks to the rapid development of mobile chipsets. For example, mobile phones nowadays have comparable computing power as computing servers a decade ago. In addition, edge servers have the potential to provide low-latency AI services for mobile users which are infeasible to be directly implemented on devices. Since edge servers have relatively less powerful computation resources than cloud centers, it is necessary to employ joint design principles across edge servers and edge devices to further reduce execution latency and enhance privacy [13]. The advances in edge computing thus provides opportunities for pushing AI frontiers from the cloud center to network edges, stimulating a new research area known as *edge AI*, including both AI model training and inference procedures.

Training at network edges is challenging and requires coordinating massive edge nodes to collaboratively build a machine learning model [19]. Each edge node usually has access to only a small subset of training data, which is the fundamental difference from traditional cloud-based model training [20]. The information exchange across edge nodes in edge training results in high communication cost, especially in the wireless environment with limited bandwidth. This brings a main bottleneck in edge training. It is interesting to note that a number of works have revisited the communication theory for addressing the communication challenges of edge training. The connection between data aggregation from distributed nodes in edge training and the in-network computation problem [21] in wireless sensor networks has been established in [22], which proposed an over-the-air computation approach for fast model aggregation in each round of training for on-device federated learning. In wireless communication systems, limited feedback [23] from a receiver to a transmitter is critical to reducing the information bits for realizing channel agile techniques that require channel knowledge at the transmitter. A connection between limited feedback in wireless communication and the quantization method was established in [24] for reducing the data transmission cost in edge training, which borrows ideas from the widely adopted Grassmannian quantization approach for limited feedback.

Edge inference, i.e., performing inference of AI models at network edges, enjoys the benefits of low-latency and enhanced privacy, which are critical for a wide range of AI applications such as drones, smart vehicles, and so on. As such, it has drawn significant attention from both academia and industry. Recently, deep learning models have been actively adopted in a number of applications to provide high-quality services for mobile users. For example, AI technologies have shown promises in healthcare [25], such as detection of heart failure [26] with recurrent neural network (RNN) and decisions about patient treatment [27] with reinforcement learning. However, deep neural network (DNN) models often have a huge number of parameters, which will consume considerable storage and computation resources. A typical example is the classic convolutional neural network (CNNs) architecture named AlexNet [2], which has over 60 million parameters. Therefore, model compression approaches [28], [29] have attracted much attention for deploying DNN models at network edges. It should also be noted that the power budget on edge devices is also limited, which stimulates research pursuits on energy-efficient processing of deep neural networks from signal processing perspective [8]. For IoT devices without enough memory to store the entire model, coding techniques shed light on the efficient data shuffling for distributed inference across edge nodes [30], [31].

B. Performance Measurements and Unique Challenges of Edge AI

The typical procedures for providing an AI service include *training* a machine learning model from data, and performing *inference* with the trained model. The performance of a machine learning model can be measured by its model

accuracy, which can potentially be improved by collecting more training data. However, training a machine learning model from massive data is time consuming. To train a model efficiently, distributed architectures are often adopted, which will introduce additional communication costs for exchanging information across nodes. The computation and communication costs grow extremely high for high-dimensional models such as deep neural networks. In addition, low-latency is also critical for inference in applications such as smart vehicles, smart drones, etc. We thus summarize the key performance measurements of edge AI in terms of **model accuracy** and **total latency**.

In the cloud center, cloud computing servers are connected with extremely high bandwidth networks and the training data is available to all nodes. Fundamentally distinct from cloud based AI, edge AI poses more stringent constraints on the algorithms and system architectures.

- **Limited resources on edge nodes:** Instead of the large amount of powerful GPUs and CPUs integrated servers at the nodes of cloud-based AI, there are often limited computation, storage, and power resources on edge devices, with limited link bandwidth among a large number of edge devices and the edge servers at base stations and wireless access points. For example, the classic AlexNet [2], which is designed for computer vision, has over 60 million parameters. With 512 Volta GPUs interconnected at the rate of 56Gbps, the Alexnet can be trained within record of 2.6 minutes in the data center of SenseTime [32]. As one of the most powerful GPUs in the world, one Volta GPU has 5,120 cores. However, nowadays the GPU on a powerful smart phone has much fewer cores. For example, the Mali-G76 GPU on Huawei Mate 30 Pro has only 16 cores, and the GPU in Apple A13 Bionic system-on-chip on iPhone 11 Pro has only 4 cores. The theoretical maximal speed envisioned in 5G is 10Gbps and the average speed is only 50Mbps.
- **Heterogeneous resources across edge nodes:** The variabilities in hardware, network, and power budget of edge nodes imply heterogeneous communication, computation, storage and power capabilities. The edge servers at base stations have much more computation, storage and power resources than mobile devices. For example, Apple Watch Series 5 can only afford up to 10 hours of audio playback,[1] and users may want to be involved in training tasks only when the devices are charged. To make things worse, edge devices that are connected to a metered cellular network are usually not willing to exchange information with other edge nodes.
- **Privacy and security constraints:** The privacy and security of AI services are increasingly vital especially for emerging high-stake applications in intelligent IoT. Operators expect stricter regulations and laws on preserving data privacy for service providers. For example, the General Data Protection Regulation (GDPR) [33] by the European Union grants users the right for data to be deleted or withdrawn. Federated learning [16], [20] becomes a particular relevant research topic for collaboratively building machine learning models while preserving data privacy. Robust algorithms and system designs are also proposed in [34], [35] for security concern against adversarial attacks during distributed edge training. It also shows promises to adopt the blockchain technique [36] and contract theory [37] to enhance the security of edge AI.

[1] https://www.apple.com/ca/watch/battery/

Enabling efficient edge AI is challenging for coordinating and scheduling edge nodes to efficiently perform a training or inference task under various physical and regulatory constraints. To provide efficient AI services, we shall jointly design new distributed paradigms for computing, communications, and learning. Note that the communication cost for cloud-based AI services may be relatively small compared with computational cost. However, in edge AI systems, the communication cost often becomes a dominating issue due to the stringent constraints. This paper will give a comprehensive survey on edge AI from the perspective of addressing communication challenges from both the algorithm level and system level.

C. Communication Challenges of Edge AI

Generally, there are multiple communication rounds between edge nodes for an edge AI task. Let L denote the total size of information to be exchanged per round, r denote the communication rate, N denote the number of communication rounds, and T denote the total computation time. Then the total latency in an edge AI system is given by

$$\text{Latency} = \underbrace{L/r \cdot N}_{\text{communication}} + \underbrace{T \cdot N}_{\text{computation}} . \tag{1}$$

For model training, iterative algorithms are often adopted which involve multiple communication rounds. The inference process often requires one round of collaborative computations across edge nodes. Therefore, to alleviate the communication overheads under resource and privacy constraints, it is natural to seek methods for reducing the number of communication rounds for training and the communication overhead per round for training and inference, as well as improving the communication rate.

From the end-to-end data transmission perspective, the information content of a message is measured in entropy that characterizes the amount of uncertainty. Based on this measure, the limit of lossless source coding is characterized by Shannon's source coding theory [38]. It provides a perfect answer to the best we can do if we only focus on "how to transmit" instead of "what to transmit" from one node to another. That is, the fundamental limit of the end-to-end communication problem has already been solved when the edge AI system and algorithm are fixed.

However, communication is not isolated in edge AI. From the learning algorithm perspective, "what to transmit" determines the required communication overhead per round and the number of communication rounds. This learning level perspective motivates the development of different algorithms to reduce the communication overhead per round and improve

the convergence rate. For instance, many gradient based algorithms have been proposed for accelerating the convergence of distributed training [39], [40]. In addition, lossy compression techniques such as quantization and pruning [28], [41] have drawn much attention recently to reduce the communication overhead per round.

Edge AI system design has also a great influence on the communication paradigm design across edge nodes. For instance, the target of communication in each round is to compute a certain function value with respect to the intermediate values at edge devices. In particular, the full gradient can be computed at a centralized node by aggregating the locally computed partial gradients at all local nodes. It is therefore better to be studied from the perspective of in-network computation [21], instead of treating communication and computation separately. For example, an over-the-air computation approach was developed in [22] for fast model aggregation in distributed model training for ferderated learning. In addition, efficient inference at network edges is closely related to computation offloading in edge computing [13], which is being extensively studied in both the communication and mobile computing communities.

D. Related Works and Our Contributions

There exist a few survey papers [9], [10], [42], [43] on edge AI. Particularly, the early works [9], [10] emphasized the differences between cloud-based AI and edge AI. Zhou *et al.* [9] surveyed the technologies of training and inference for deep learning models at network edges. Park *et al.* [10] focused on the opportunities of utilizing edge AI for improving wireless communication, as well as realizing edge AI over wireless channels. Murshed *et al.* [42] mainly discussed different machine learning models and neural network models, different practical applications such as video analytics and image recognition, as well as various machine learning frameworks for enabling edge AI. Deng *et al.* [43] further considered the convergence of edge computing and deep learning, i.e., the deep learning techniques for edge computing, as well as edge computing techniques for deep learning.

Unlike existing survey papers [9], [10], [42], [43], we shall present a comprehensive coverage to address the communication challenges for realizing AI at network edges. Edge AI is far from a trivial task of merely adopting the same computation and communication techniques in the cloud center. It requires learning performance aware joint design of computation and communication. Both distributed learning algorithms and distributed computing system architectures shall be customized according to the considered AI model, data availability, and the heterogeneous resources at edge nodes for reducing communication overheads during training and inference. We summarize the research topics on edge AI as algorithm-level designs and system-level designs, which are listed more specifically as follows:

- **Algorithm level:** At the algorithm level, the communication rounds of training a model can be reduced by accelerating convergence, while communication overhead per round can be reduced by information compression techniques (e.g., sparsification, quantization, etc.). We first survey different types of edge AI algorithms including the zeroth-order, first-order, second-order and federated optimization algorithm, as well as their applications in edge AI. For example, in the context of reinforcement learning, model-free based methods turn the reinforcement learning problem into zeroth-order optimization [44]. Although first-oder methods are widely used in DNNs training, second-order methods and federated optimization become appealing in edge AI given the growing computational capabilities of devices. As we can see from Fig. 2, the algorithm closer to the right side can potentially achieve better accuracy with less communication rounds, at the cost of more computation resources per round. Note that we list federated optimization methods separately due to its unique motivation to protect private data at each node. For each type of algorithms, there are a number of works focusing on further reducing communication cost. We give a comprehensive survey on the algorithm level in Section III to address the communication challenges in edge AI.
- **System level:** From the system perspective, data distribution (e.g., distributed across edge devices), model parameters (e.g., partitioned and deployed across edge devices and edge servers), computation (e.g., MapReduce), and communication mechanisms (e.g., aggregation at a central node) can be diverse in different applications. There are two main edge AI system architectures for training, i.e., the data partition system and model partition system, based on the availability of data and model. After training the AI model, model deployment is critical for achieving low-latency AI services. There are also other general edge computing paradigms in edge AI systems that address the trade-off between computation and communication via coding techniques. There are different types of communication problems arising from the deployment of machine learning algorithms on different system architectures, which typically involve the distributed mode and decentralized mode depending on the existence of a central node. We shall survey various system-level approaches to achieve efficient communications in Section IV.

We summarize the main topics and highlighted technologies included in this paper in Table II.

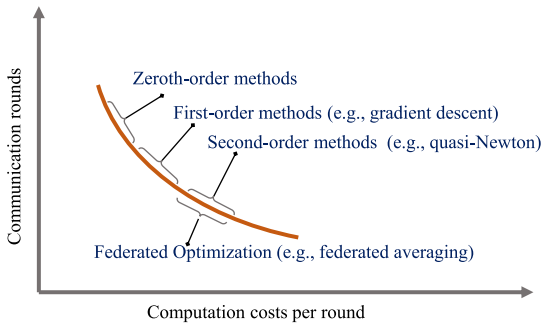


Fig. 2. Communication rounds and computation costs per round for different types of training algorithms. The trade-off has been widely used in designing algorithms, by performing more computation at each round in exchange for fewer number of communication rounds till convergence.

TABLE II
AN OVERVIEW OF TOPICS COVERED IN THE PAPER

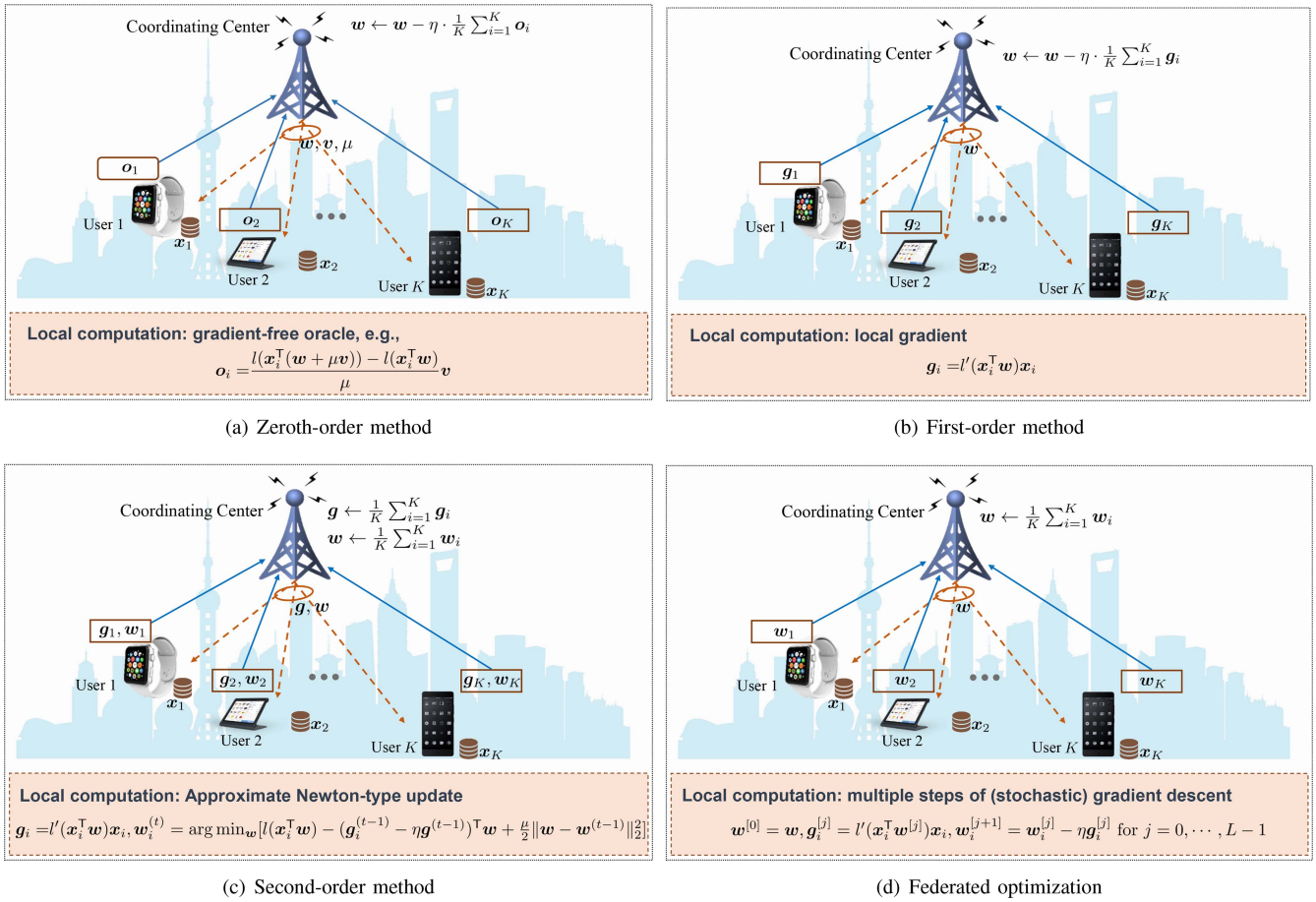
Category	Topic	Representative Results
Section III: Communication-Efficient Algorithms for Edge AI	Section III-A: Zeroth-Order Methods	• Optimal rates for zeroth-order convex optimization [45] • Distributed zeroth-order algorithms over time-varying networks [46], [47]
	Section III-B: First-Order Methods	• Variance reduction for minimizing communication rounds [39], [40] • Gradient reuse for minimizing communication bandwidth [48], [49] • Relating gradient quantization to limited feedback in wireless communication [24] • Communicating only important gradients for minimizing communication bandwidth [50], [41]
	Section III-C: Second-Order Methods	• Stochastic quasi-Newton methods [51], [52], [53] • Approximate Newton-type methods [54], [55], [56], [57]
	Section III-D: Federated Optimization	• Federated averaging algorithm [19] and dual coordinate ascent algorithm [58] for minimizing communication rounds • Handling the system and statistical heterogeneity of distributed learning [59], [60] • Compressing DNN models with vector quantization [61], binary weights [62], [63], [64], randomized sketching [65], [66], [67], network pruning [68], [69], [70], [28], [71], [72], [73], [74], sparse regularization [75], [76], [77], and structural matrix designing for minimizing communication bandwidth [78], [79], [80], [81], [82], [83], [84], [85], [86]
Section IV: Communication-Efficient Edge AI Systems	Section IV-B: Data Partition Based Edge Training Systems	• Fast aggregation via over-the-air computation [22], [87], [88], [89] • Aggregation frequency control with limited bandwidth and computation resources [90], [91], [92] • Data reshuffling via index coding and pliable index coding for improving training performance [93], [94], [95] • Straggler mitigation via coded computing [96], [97], [98], [99], [100], [101], [102], [103], [104] • Training in decentralized system mode [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115]
	Section IV-C: Model Partition Based Edge Training Systems	• Model partition across a large number of nodes to balance computation and communication [116], [117], [118] • Model partition across edge device and edge server to avoid the exposure of users data [119], [120] • Vertical architecture for privacy with vertically partitioned data and model [20], [121], [122], [123], [124], [125], [126]
	Section IV-D: Computation Offloading Based Edge Inference Systems	Server-based edge inference: • Partial data transmission for communication-efficient inference [127], [128], [129], [130] • Raw data encoding for communication-efficient inference [131], [132] • Cooperative downlink transmission for communication-efficient inference [133], [134] Device-edge joint inference: • Early exit: [135], [136] • Encoded transmission and pruning for compressing the transmitted data [137], [138] • Coded computing for cooperative edge inference [139]
	Section IV-E: General Edge Computing Systems	• Coding techniques for efficient data shuffling [140], [141], [30], [31], [142], [143], [144], [145], [146] • Coding techniques for straggler mitigation [145], [147], [148], [149]

III. COMMUNICATION-EFFICIENT ALGORITHMS FOR EDGE AI

Distributed machine learning has been mainly investigated in the environment with abundant computing resources, large memory, and high-bandwidth networking, e.g., in cloud data centers. The extension to the edge AI system is highly non-trivial due to the isolated data at distributed mobile devices, limited computing resources, and the heterogeneity in communication links. Communication-efficient methods will be critical to exploit the distributed data samples and utilize various available computing resources for achieving excellent learning performance. This section introduces communication-efficient approaches for edge AI at the algorithmic level, including zeroth-order methods, first-order methods, second-order methods, and federated optimization. As illustrated in Fig. 2, these methods achieve different trade-offs among the local computation and communication cost. Fig. 3 provides illustrations of local operations and communication messages of different methods.

A. Communication-Efficient Zeroth-Order Methods

Zeroth-oder (derivative-free) methods [150] are increasingly adopted in the applications where only the function value is



(a) Zeroth-order method

(b) First-order method

(c) Second-order method

(d) Federated optimization

Fig. 3. Illustration of different optimization methods for model training. As a typical example, a generalized linear model is trained where each node i has one data instance $\boldsymbol{x}_i$. That is, the target is to optimize $\min_{\boldsymbol{w}} \frac{1}{K}\sum_{i=1}^{K} l(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i)$, where l is the loss function. The first-order derivative of function ℓ is denoted as ℓ'. (a) Zeroth-order method: only the function value can be evaluated during training [150]. (b) First-order method: gradient descent. (c) Second-order method: DANE [54]. (d) Federated optimization: federated averaging algorithm [19].

available, but the derivative information is computationally difficult to obtain, or is even not well defined. For the distributed setting with a central coordinating center shown in Fig. 3(a), only a function value scalar is required to be transmitted to the central node in uplink transmission. In the field of reinforcement learning, zeroth-order methods have been widely used for policy function learning without ever building a model [44]. Zeroth-order methods have also been adopted to black-box adversarial attacks on DNNs since most real world systems do not release their internal DNNs structure and weights [151].

In zeroth-order optimization algorithms, the full gradients are typically estimated via gradient estimators based on only the function values [152]. For instance, we can use the quantity $(l(\boldsymbol{w}+\mu\boldsymbol{v}) - l(\boldsymbol{w}))\boldsymbol{v}/\mu$ to approximate the gradient of function $l(\boldsymbol{w})$ at point $\boldsymbol{w}$. It was shown in [45] that this kind of derivative-free algorithm only suffers a factor of at most $\sqrt{d}$ in the convergence rate over traditional stochastic gradient methods for d-dimensional convex optimization problems. Under time-varying random network topologies, recent studies [46], [47] have investigated the distributed zeroth-order optimization algorithms for unconstrained convex optimization in multi-agent systems. Convex optimization with a set of convex constraints have been studied in [153], [154]. Nonconvex multi-agent optimization has been studied in [155] for different types of network topologies, including undirected connected networks or star networks under the setting where the agent can only access the values of its local function.

To develop communication-efficient distributed zeroth-order optimization methods, there have been a number of works on reducing the number of per-device communication. For instance, it was proposed in [156] that at each iteration each device communicates with its neighbors with some probability that is independent from others and the past, and this probability parameter decays to zero at a carefully tuned rate. For such a distributed zeroth-oder method, the convergence rate of the mean squared error of solutions is established in terms of the communication costs, i.e., the number of per-node transmissions to neighboring nodes in the network, instead of the iteration number. The subsequent work [157] improved the convergence rate under additional smoothness assumptions. Quantization techniques are also adopted to reduce the communication cost per communication round. The paper [158] considered a distributed gradient-free algorithm for multi-agent convex optimization, where the agents can only exchange quantized data information due to limited bandwidth. In the extreme case considered in [159], each

estimated gradient is further quantized into 1 bit, which enjoys high communication efficiency in distributed optimization scenarios.

B. Communication-Efficient First-Order Methods

First-order optimization methods are the most commonly used algorithms in machine learning, which are mainly based on gradient descent methods as shown in Fig. 3(b). The idea of gradient descent methods is to iteratively update variables in the opposite direction of the gradients of the loss function at that point with an appropriate step size (a.k.a., a learning rate). As the computational complexity at each iteration scales with the number of data samples and the dimension of the model parameter, it is generally infeasible to train large machine learning models with tremendous amount of training data samples on a single device. Therefore, distributed training techniques have been proposed to mitigate the computation cost, with additional communication costs. Meanwhile, as the training dataset becomes larger and larger, stochastic gradient descent (SGD) method emerges as an appealing solution, in which only one training sample is used to compute the gradient at each iteration. In edge AI systems with inherently isolated data, distributed realizations of SGD will play a key role and should be carefully investigated.

To apply first-order methods in large-scale distributed edge AI systems, the substantial demand for communication among devices for gradient exchange is one of the main bottlenecks. One way to address this issue is to reduce the communication round by accelerating the convergence rate of the learning algorithms. Another approach is to reduce the communication overhead per round, which includes gradient reuse method, quantization, sparsification, and sketching based compression methods. These two approaches are elaborated in the following.

1) Minimizing Communication Round: We first consider an extreme case. The distributed optimization approach with the minimum communication round, i.e., only one communication round, is that each device performs independent optimization. For example, each node adopts SGD to compute local model parameters, and a server then averages these model parameters in the end. As shown in [160], the overall run time decreases significantly as the number of devices increases for some learning tasks. Subsequently, it was shown in [161] that this one-round communication approach can achieve the same order-optimal sample complexity in terms of mean-squared error of model parameters as the centralized setting under a reasonable set of conditions. The order-optimal sample complexity can be obtained by performing a stochastic gradient-based methods on each devices [161]. However, one round communication restricts the ability to exchange information during training, which is in general not sufficient for training large models (e.g., DNNs) to achieve the target accuracy in practice.

In general settings where devices upload their local gradients to a fusion center at each iteration, it is critical to reduce the communication round by accelerating the convergence rate of the algorithm. Shamir and Srebro [162] proposed to accelerate mini-batch SGD by using the largest possible mini-batch size that does not hurt the sample complexity, and it shows that the communication cost decreases linearly with the size of the mini-batch. Yuan *et al.* [39] proposed an amortized variance-reduced gradient algorithm for a decentralized setting, where each device collects data that is spatially distributed and all devices are only allowed to communicate with direct neighbors. In addition, a mini-batch strategy is adopted by [39] to achieve communication efficiency. However, it has been shown in [163], [164] that too large mini-batch sizes will result in a degradation in the generalization of the model. In practice, additional efforts should be taken to reduce this generalization drop. For instance, it was shown in [165] that training with large minibatch sizes up to 8192 images achieves the same accuracy as small mini-batch settings by adjusting learning rates as a function of mini-batch size. This idea was also adopted by [166] to train DNNs for automatic speech recognition tasks using large batch sizes in oder to accelerate the total training process.

The statistical heterogeneity of data hinders the fast convergence of first-order algorithms. To address this issue, there have been lots of efforts. Arjevani and Shamir [167] studied the scenarios where each device has access to a different subset of data to minimize the averaged loss function over all devices. They established a lower bound on the rounds of communication, which is shown to be achieved by the algorithm of [168] for quadratic and strongly convex functions. But how to design optimal algorithms in terms of communication efficiency for general functions remains an open problem. By utilizing additional storage space of devices, Lee *et al.* [40] proposed to assign two subsets of data to each device. The first subset is from a random partition and the second subset is randomly sampled with replacement from the overall datasets. Since each device has access to both data subsets, the authors proposed a distributed stochastic variance reduced gradient method to minimize the communication round, in which the batch gradients are computed in parallel on different devices and the algorithm utilizes the local data sampled with replacement to construct the unbiased stochastic gradient in each iterative update. For non-convex optimization problems, Garber *et al.* [169] proposed a stochastic distributed algorithm to solve the principal component analysis problem, which gives considerable acceleration in terms of communication rounds over previous distributed algorithms.

2) Minimizing Communication Bandwidth: Another series of works focus on reducing the size of local updates from each device, thereby reducing the overall communication cost. In the following, we review three representative techniques, i.e., gradient reuse, gradient quantization, and gradient sparsification.

• **Gradient reuse:** To minimize a sum of smooth loss functions distributed among multiple devices, considering that the gradients of some devices vary slowly between two consecutive communication rounds, a lazily aggregated gradient (LAG) method was proposed by [48] which uses outdated gradients of these devices at the fusion center. Specifically, theses devices upload nothing during this communication round,

which is able to reduce communication overheads per round significantly. Theoretically, it was shown in [48] that LAG achieves the same order of convergence rates as the batch gradient descent method under the cases where the loss functions are strongly-convex, convex, or nonconvex smooth. If the distributed datasets are heterogeneous, LAG can achieve a target accuracy with considerably less communication costs measured as the total number of transmissions over all the devices in comparison with the batch gradient descent method. In addition, a similar gradient reuse idea was adopted in distributed reinforcement learning to achieve communication efficient training [49].

• **Gradient quantization:** To reduce the communication cost of gradient aggregation, some scalar quantization methods have been proposed to compress the gradients by a small number of bits instead of using floating-point representation. To estimate the mean of the gradient vectors collected from devices, Suresh *et al.* [170] analyzed the mean squared error for several quantization schemes without probabilistic assumptions on the data from the information theoretic perspective. In the view of distributed learning algorithms. Alistarh *et al.* [171] has investigated the quantized stochastic gradient descent (QSGD) to study the trade-off between communication costs and convergence guarantees. Specifically, each device can adjust the number of bits sent per iteration according to the variance added to the device. As demonstrated in [171], each device can transmit no more than $2.8n + 32$ bits per iteration in expectation, where n is the number of model parameters, while the variance is only increased by a factor of 2. Compared to full precision SGD, using approximately $2.8n$ bits of communication per iteration as opposed to $32n$ bits will only result in at most $2\times$ more iterations, which leads to bandwidth savings of approximately $5.7\times$. For distributed training the shallowest neural networks consisting of a single rectified linear unit, it was shown in [172] that the quantized stochastic gradient method converges to the global optima at a linear convergence rate. Seide *et al.* [173] proposed to quantize the gradient using only one bit, achieving a 10 times speed-up on distributed training of speech DNNs with a small accuracy loss. Theoretically, Bernstein *et al.* [174] provided rigorous analysis for the sign-based distributed stochastic gradient descent algorithm, where each device sends the sign information of the gradients to a fusion center and the sign information of the aggregated gradients signs is returned to each device for updating model parameters. This scheme is shown to achieve the same reduction in variance as full precision distributed SGD and converge to a stationary point of a general non-convex function.

As pointed out in [175], [176], scalar quantization methods fail under decentralized networks without a central aggregation node. To address this issue, extrapolation compression and difference compression methods were proposed in [175], and a gradient vector quantization technique was proposed in [176] to exploit the correlations between CNN gradients. Vector quantization [177] by jointly quantizing all entries of a vector can achieve the optimal rate-distortion trade-off, which, however, comes at the price of high complexity that increases with the vector length. Interestingly, it was found in [24] that Grassmannian quantization, a vector quantization method that has already been widely adopted in wireless communication for limited feedback, can be applied for gradient quantization. Limited feedback is an area of studying efficient feedback of quantized vectors from a receiver to a transmitter for channel adaptive transmission schemes since the communication cost of feedback is extremely high in massive MIMO communication systems. This motivated [24] to develop an efficient Grassmannian quantization scheme for high-dimensional gradient compression in distributed learning.

Additionally, Jiang *et al.* [178] proposed to use quantile sketch, a non-uniform quantization method for gradient compression. Sketch is a technique of approximating input data with a probablistic data structure. In [178], the gradient values are summarized into a number of buckets, whose indices are further encoded by a binary representation since the number of buckets is relatively small.

• **Gradient sparsification:** The basic idea behind gradient sparsification is to communicate only important gradients according to some criteria. This is based on the observation that many gradients are normally very small during training. Strom [179] proposed to leave out the gradients below a predefined constant threshold. Chen *et al.* [50] proposed AdaComp via localized selection of gradient residues, which automatically tunes the compression rate depending on local activity. It was demonstrated that AdaComp can achieve a compression ratio of around $200\times$ for fully-connected layers and $40\times$ for convolutional layers without noticeable degradation of top-1 accuracy on ImageNet dataset. Deep gradient compression was proposed in [41] based on a gradient sparsification approach, where only gradients exceeding a threshold are communicated, while the remaining gradients are accumulated until they reach the threshold. Several techniques including momentum correction, local gradient clipping, momentum factor masking, and warm-up training are adopted to preserve the accuracy. This deep gradient compression approach is shown to achieve a gradient compression ratio from $270\times$ to $600\times$ without losing accuracy for a wide range of DNNs and RNNs [41]. In [180], to ensure the sparsified gradient to be unbiased, the authors proposed to drop some coordinates of the stochastic gradient vectors randomly and amplify the rest of the coordinates appropriately. For both convex and non-convex smooth objectives, under analytic assumptions, it was shown in [181] that sparsifying gradients by magnitude with local error correction provides convergence guarantees. Thus, providing a theoretical foundation for numerous empirical results on training large-scale recurrent neural networks on a wide range of applications.

C. Communication-Efficient Second-Order Methods

First-order algorithms only require the computation of gradient-type updates, and thus reduce the amount of local computation at each device. But the main drawback is that the required number of communication rounds is still huge due to the slow convergence rate. It thus motivates to exploit second-order curvature information into distributed learning algorithms to improve the convergence rate for edge training.

However, exact second-order methods require the computation, storage and even communication of a Hessian matrix, which results in tremendous overhead. Therefore, one has to resort to approximate methods such as illustrated in Fig. 3(c) [54]. The works on communication-efficient second-order methods can be categorized into two types. One is to maintain a global approximated inverse Hessian matrix in the central node, and the other line of works propose to solve a second-order approximation problem locally at each device.

A common approach to develop approximate second-order methods is to take the merits of the well-known quasi-Newton method, namely Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) [182], which avoids the high cost of computing the inversion of Hessian matrix via directly estimating the inverse Hessian matrix. In learning with large amounts of training data, it is a critical problem to develop a mini-batch stochastic quasi-Newton method. However, directly extending L-BFGS to a stochastic version does not result in a stable approximation of the inverse Hessian matrix. Schraudolph *et al.* [51] developed a stochastic L-BFGS for online convex optimization without line search, which is often problematic in a stochastic algorithm. But there may be a high level of noise in its Hessian approximation. To provide stable and productive Hessian approximations, Byrd *et al.* [52] developed a stochastic quasi-Newton method by updating the estimated inverse Hessian matrix every L iterations using sub-sampled Hessian-vector products. The inverse Hessian matrix maintained in a central node is updated by collecting only a Hessian-vector product update at each device. Moritz *et al.* [53] proposed a linearly convergent stochastic L-BFGS algorithm via obtaining a more stable and higher precision estimation of the inverse Hessian matrix, but it requires higher computation and communication overhead at each round.

Another main idea of communication-efficient second-order methods is to solve a second-order approximation problem at each device without maintaining and computing a global Hessian matrix. To reduce the communication overhead at each round, Shamir *et al.* [54] proposed a distributed approximate Newton-type method named as "DANE" by solving an approximate local Newton system at each device with a global aggregation step, which only requires the same communication bandwidth as first-order distributed learning algorithms. Subsequently, the algorithm "DiSCO" proposed in [55] solved a more accurate second-order approximation at per communication round by approximately solving the global Newton system with a distributed preconditioned conjugate gradient method. It reduces the communication rounds compared with "DANE", while the computation cost at the master machine grows roughly cubically with the model dimension. Wang *et al.* [56] proposed an improved approximate Newton method "GIANT" to further reduce the communication round via conjugate gradient steps at each device, which is shown to outperform "DANE" and "DiSCO". Note that the communication of these approaches involves the transmission of a global update to each device and the aggregation of local update from each device at per round, both with the same size as the number of model parameters. However, the convergence results of "DANE", "DiSCO", and "GIANT" require a high accuracy solution to the subproblem at each device. An adaptive distributed Newton method was proposed in [57] by additionally transmitting a scalar parameter accounting for the information loss of distributed second-order approximation at per round, which outperforms "GIANT" in numerical experiments.

D. Communication-Efficient Federated Optimization

In the edge training system, the local dataset at each device is usually only a small subset of the overall dataset. Furthermore, the rapid advancement of CPUs and GPUs on mobile devices makes the computation essentially free in comparison to the communication cost. Thus, a natural idea is to use additional local computation to decrease the communication cost. Federated optimization [16] is a framework of iteratively performing a local training algorithm (such as multiple steps of SGD as illustrated in Fig. 3(d)) based on the dataset at each device and aggregating the local updated models, i.e., computing the average (or weighted average) of the local updated model parameters. This framework provides additional privacy protection for data, and has the potential of reducing the number of communication rounds for aggregating updates from a large number of mobile devices. The concern of data privacy and security is becoming a worldwide major issue, especially for emerging high-stake applications in intelligent IoT, which prompted governments to enact new regualtions such as General Data Protection Regulation (GDPR) [33]. There are a line of works studying federated optimization algorithms [19], [58], [60] to reduce the communication rounds. In addition, a number of model compression methods have been proposed to reduce the model size, either during the local training process or compressing the model parameters after local training, which can further reduce the communication cost of aggregation for federated optimization [183]. These methods are reviewed in this part.

1) Minimizing Communication Round: Jaggi *et al.* [58] proposed a framework named "CoCoA" by leveraging the primal-dual structure of a convex loss function of general linear models with a convex regularization term. In each communication round, each mobile device performs multiple steps of a dual optimization method based on local dataset in exchange for fewer communication rounds, followed by computing the average of updated local models. Motivated by [58], authors in [60] further proposed a communication-efficient federated optimization algorithm called "MOCHA" for multi-task learning. By returning an additional accuracy level parameter, it is also capable of dealing with straggling devices. However, these algorithms are not suitable for a general machine learning problem when the strong duality fails or the dual problem is difficult to obtain.

The Federated Averaging (FedAvg) [19] algorithm is another communication-efficient federated optimization algorithm by updating local model at each device with a given number of SGD iterations and model averaging. It is generalized from the traditional one-shot averaging algorithm [184] that is applicable only when the data samples at each device are drawn from the same distribution. In per round of communication, each device performs a given number of steps

of SGD with a global model as the initial point, and the aggregated global model is given by the weighted average of all local models. The weights are chosen as the sizes of the local training dataset, which is shown to be robust to not independently and identically distributed (non-IID) data distribution and unbalanced data across mobile devices. Wang and Joshi [185] provided the convergence result of the FedAvg algorithm to a stationary point. To reduce the costly communication with the remote cloud, edge server assisted hierarchical federated averaging was proposed in [186]. By exploiting the highly efficient local communications with edge servers, it achieves significant training speedup compared with the cloud-based approach. With infrequent model aggregation at the cloud, it also achieves higher model performance than edge-based training, as data from more users can be accessed.

For the FedAvg algorithm, the steps of local SGD at each device should be chosen carefully given the existence of statistical heterogeneity, i.e., when the local data across devices are non-IID. If too many steps of SGD are performed locally, the learning performance will be degraded. To address this problem, the FedProx algorithm [59] was proposed by adding a proximal term in the local objective function to restrict the local updated model to be close to the global model, instead of initializing each local model with the global updated at each communication round. Its convergence guarantees are also provided via characterizing the heterogeneity with a device dissimilarity assumption. Numerical results demonstrate that FedProx is more robust to the statistical heterogeneity across devices. In addition, Chen *et al.* [187] optimized device selection and wireless resource allocation for federated learning model aggregation over a wireless channel, and derived the convergence rate.

2) Minimizing Communication Bandwidth: Transmitting the model parameters per communication round generally results in a huge communication overhead since the number of model parameters can be very large. Therefore, it is important to reduce the model size to alleviate the communication overhead [183]. To this end, model compression is one of the promising approaches. We survey the main techniques adopted in model compression in this subsection.

• **Quantization:** Quantization compresses DNNs by representing the weights by fewer bits instead of adopting the 32-bit floating point format. The works [61], [188] adopt *k*-means clustering to the weights of a pre-trained DNN. At the training stage, it has been shown that DNNs can be trained using only 16-bit wide fixed-point number representation by stochastic rounding [189], which induces little to no degradation in the classification accuracy. In the extreme case, the weights are represented by 1-bit, but the naive approach that binarizes pre-trained DNNs directly shall bring performance loss significantly. Therefore, the main idea behind binarization is to learn the binary weights or activation during training, which are thoroughly investigated in [62], [63], [64]. This kind of method allows a substantial computational speedup on devices due to the bit-wise operations. It may also reduce the communication cost in federated learning significantly as the weights are represented by 1-bit.

• **Sketching:** Randomized sketching [190], [191] is a powerful tool for dimensionality reduction, which can be applied to model compression. In [65], HashedNet sketches the weights of neural networks using a hash function, and enforces all weights that are mapped to the same hash bucket to share a single parameter value. But it is only applicable to fully connected neural networks. The subsequent work [66] extended it to CNNs, which is achieved by first converting filter weights to the frequency domain and then grouping the corresponding frequency parameters into hash buckets using a low-cost hash function. Theoretically, it was shown in [67] that such Hashing-based neural networks have nice properties, i.e., local strong convexity and smoothness around the global minimizer.

• **Pruning:** Network pruning generally compresses DNNs by removing the connections, filters or channels according to some criteria. Early works include the Optimal Brain Damage [68] and the Optimal Brain Surgeon [69], which proposed to remove the connections between neurons based on the Hessian of the loss function given a trained DNN. Recently, a line of research is to prune redundant, less important connections in a pre-trained DNN. For instance, the work in [70] proposed to prune the unimportant weights of a pre-trained network and retrain the network to fine tune the weights of the remaining connections, which reduces the number of parameters of AlexNet by $9\times$ without harming the accuracy. Deep compression was proposed in [28] to compress DNNs via three stages, i.e., pruning, trained quantization and Huffman coding, which yields considerably compact DNNs. For example, the storage size of AlexNet is reduced by $35\times$ on the ImageNet dataset without loss of accuracy. From a Bayesian point of view, network pruning was also investigated in [71], [72]. However, such heuristic methods present no convergence guarantees. Instead, Aghasi *et al.* [73] proposed to prune the network layer-by-layer via convex programming, which also shows that the overall performance drop can be bounded by the sum of the reconstruction error of each layer. Subsequently, iterative reweighed optimization has been adopted to further prune the DNN with convergence guarantees [74].

• **Sparse regularization:** There is a growing interest in learning compact DNNs without pre-training, which is achieved by adding regularizers to the loss function during training in order to induce sparsity in DNNs. In [75], the authors proposed to use a regularizer based on $\ell_{2,1}$-norm to induce group-sparse structured convolution kernels when training CNNs, which leads to computational speedups. To remove trivial filters, channels and even layers at the training stage, the work in [76] proposed to add structured sparsity regularization on each layer. Theoretically, the convergence behavior of gradient descent algorithms for learning shallow compact neural networks was depicted in [77], which also shows the required sample complexity for efficient learning.

• **Structural matrix designing:** The main idea behind low-rank matrix factorization approaches for compressing DNNs is to apply low-rank matrix factorization techniques to the weight matrix of DNNs. For a low rank matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $\text{rank}(\boldsymbol{A}) = r$, we can represent it as $\boldsymbol{A} = \boldsymbol{BC}$ where $\boldsymbol{B} \in \mathbb{R}^{m\times r}$. Therefore, we reduce the total parameters from mn to $mr + nr$, which is able to reduce the computational complexity and

storage. For example, the work in [78] showed that the number of parameters of the DNNs can be reduced by 30%−50% for large vocabulary continuous speech recognition tasks via low-rank matrix factorization of the final weight layer. In [79], in order to accelerate convolution, each convolutional layer is approximated by a low-rank matrix, and different approximation metrics are studied to improve the performance. The work in [80] proposed to speed up the convolutional layers by constructing a low rank basis of rank-one filters for a pre-trained CNN.

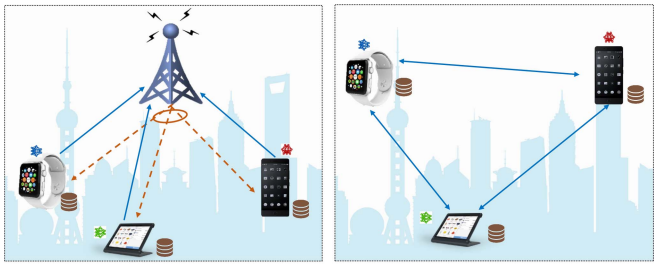
Low-rank methods have also been exploited at the training stage. In [81], low-rank methods were exploited to reduce the number of network parameters that are learned during training. Low-rank methods have also been adopted to learn separable filters to accelerate convolution in [82], [83], which is achieved by adding additional regularization to find low-rank filters.

Besides low-rank matrix factorization, another way to reduce the number of parameters of weight matrix is leveraging structured matrices which can describe $m \times n$ matrices with much fewer parameters than mn. In this way, Sindhwani *et al.* [84] proposed to learn structured parameter matrices of DNNs, which also accelerates inference and training dramatically via fast matrix-vector products and gradient computation. The work in [85] proposed to impose the circulant structure on the weight matrix of fully-connected layers to accelerate computation both at training and inference stages. In [86], the authors presented an adaptive Fastfood transform to reparameterize the matrix-vector multiplication of fully-connected layers, thereby reducing the storage and computational costs.

As discussed in this section, there are two main aspects in designing communication-efficient algorithms for edge AI. That is, the communication rounds required by an algorithm (which could be reduced by improving the convergence rate), and the communication bandwidth required by each communication round (which could be reduced by adopting a number of data compression techniques). There are often other concerns/constraints in various real world applications (such as the need of encryption, a constraint on the maximum number of data exchanges), for which researchers can customize their algorithms based on the general algorithm frameworks that have been introduced in this section. For example, according to the data availablity constraint and privacy concerns in vertical federated learning, Yang *et al.* [192] designed a communication-efficient algorithm based on the stochastic quasi-Newton method proposed in [52] and the Homomorphic encryption technique.

IV. Communication-Efficient Edge AI Systems

Due to the limited computation, storage, and communication resources of edge nodes, as well as the privacy, security, low-latency, and reliability requirements of AI applications, a variety of edge AI system architectures have been proposed and investigated for efficient training and inference. This section gives a comprehensive survey of different edge AI systems and topics therein. It starts with a general discussion on different architectures, and then introduces them one by one.



(a) Distributed mode (b) Decentralized mode

Fig. 4. Two different types of edge training systems.

A. Architectures of Edge AI Systems

We summarize the main system architectures of edge AI into four categories. According to the availability data and model parameters, data partition based edge training systems and model partition based edge training systems are two common system architectures for efficiently training at network edges. To achieve low-latency inference, computation offloading based edge inference systems is a promising approach by offloading the entire or a part of inference tasks from resource limited edge devices to proximate edge servers. There are also edge AI systems defined by general computing paradigms, which can be termed as general edge computing systems.

• **Data partition based edge training systems:** For data partition based edge training systems, the data is massively distributed over a number of edge devices, and each edge device has only a subset of the whole dataset. Then the edge AI model can be trained by pooling the computation capabilities of edge devices. During training, each edge device holds a replica of the complete AI model to compute a local update. This procedure often requires a centralized coordinating center, e.g., an edge server, for scheduling a number of edge devices, aggregating the local updates from edge devices, etc. There are also works considering decentralized systems where edge devices communicate with each other directly. Edge training systems with a central node are usually called *distributed* system modes, while systems without a central node are called *decentralized* system modes, as demonstrated in Fig. 4.

• **Model partition based edge training systems:** In model partition based edge training systems, each node does not have replica of all the model parameters, i.e., the AI model is partitioned and distributed across multiple nodes. Model partition is needed when very deep machine learning models are applied. Some works proposed to balance computation and communication overhead via model partition for accelerating the training process. Furthermore, model partition based edge training systems garner much attention for preserving the data privacy during training when each edge node can only access to partial data attributes for a common set of user identities. It is often referred to as *vertical federated learning* [20]. To preserve data privacy, it is proposed to train a model through the synergy of the edge device and edge server by performing simple processing at the device and uploading the intermediate values to a powerful edge server. This is realized by deploying a small part of model parameters on the device and the remaining part on the edge server to avoid the exposure of users' data.

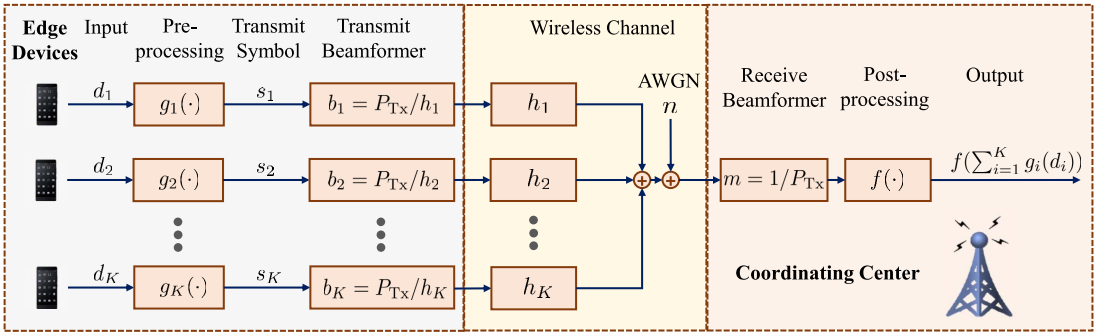


Fig. 5. Illustration of fast aggregation via over-the-air computation. We exemplify it with a simple single-antenna system for computing $f(\sum_{i=1}^{K} g_i(d_i))$ from K distributed edge devices.

• **Computation offloading based edge inference systems:** To enable low-latency edge AI services, it is critical to deploy the trained model proximate to end users. Unfortunately, it is often infeasible to deploy large models, especially DNN models, directly on each device for local inference due to the limited storage, computation and battery resources. Therefore, a promising solution is to push the AI model and massive computations to proximate edge servers, which prompts the recent proposal of computation offloading based edge inference systems [13]. We divide the works on computation offloading based edge inference systems into two classes, i.e., deploying the entire model on an edge server, and partitioning the model and deploying across the edge device and edge server.

• **General edge computing systems:** Beyond the systems mentioned above, there are also edge AI systems defined by general computing paradigms, e.g., MapReduce [193]. The MapReduce-like frameworks often consider distributed data input and distributed model deployment jointly for accelerating distributed training or inference. In such systems, reducing the communication overhead for data shuffling between multiple nodes becomes a critical task. Interestingly, coding technique plays a critical role in scalable data shuffling [31], [141] as well as straggler mitigation [145].

In the remaining part of this section, we discuss the important topics and involved techniques that address the communication challenges for these system architectures.

B. Data Partition Based Edge Training Systems

In data partition based edge training systems, each device usually has a subset of the training data and a replica of the machine learning model. The training can be accomplished by performing local computation and periodically exchanging local updates from mobile devices. The main advantage of such a system is that it is applicable to most of the model architectures and scales well. The main drawback is that the model size and the operations that are needed to complete the local computation are limited by the storage size and computation capabilities of each device. In the following, we separately discuss distributed and decentralized system modes.

1) Distributed System Mode: In the distributed system mode, each edge device computes a local update according to its local data samples, and the central node shall periodically aggregate local updates from edge devices. The communication bottleneck comes from aggregating the local updates from mobile devices and straggler devices. The efforts for addressing the communication challenges in distributed data partition based training systems are listed as follows:

• **Fast aggregation via over-the-air computation:** Over-the-air computation is an efficient approach to compute a function of distributed data by exploiting the signal superposition property of the wireless multiple access channel [194]. As shown in Fig. 5, we are able to jointly consider communication and computation to reduce the communication costs significantly. In particular, the function that can be computable via over-the-air computation is called the nomographic function [195]. In [196], over-the-air computation based on MIMO, i.e., multi-antenna techniques, is adopted in high-mobility multi-modal sensing for fast aggregation, where the receive beamforming is designed based on the differential geometry approach. A blind over-the-air computation approach was further developed in [197] to compute the desired functions without the knowledge of channel state information. In distributed machine learning, we first compute the local updates (e.g., gradients and model parameters) at each worker, and aggregate these values over the wireless channel. For aggregation functions that fall into the class of nomographic functions, we are able to improve the communication efficiency by exploiting over-the-air computing. It should be noted that digital modulation schemes for over-the-air computation are advocated in [198], [199], [200], [201] due to its easier implementation on the existing communication systems and its less stringent requirement of synchronization compared to anolog schemes.

To improve the communication efficiency for federated learning, a number of parallel works [22], [87], [88] proposed to adopt the over-the-air computation approach for fast model aggregation instead of the traditional communication-and-computation separation method. This is motivated by the fact that the aggregating function is a linear combination of updates from distributed mobile devices, which falls into the set of nomographic functions. Using transceiver design by exploring the signal superposition property of a wireless multiple access channel, over-the-air computation can improve the communication efficiency and reduce the required bandwidth. In

particular, the joint device selection and beamforming design problem was considered in [22], for which sparse and low-rank optimization methods were proposed, yielding admirable performance of the proposed over-the-air computation for fast model aggregation. Zhu *et al.* [87] characterized two trade-offs between communication and learning performance for over-the-air computation based fast aggregation in federated edge learning. The first one is the trade-off between the updated quality measured by the receive SNR and the truncation ratio of model parameters due to the proposed truncated-channel-inversion policy for deep fading channels. The second one is the trade-off between the receive SNR and the fraction of exploited data, namely, the fraction of scheduling cell-interior devices if the data distributed over devices uniformly. Based on over-the-air computation, Amiri and Gunduz [88] proposed a gradient sparsification and random linear projection method to reduce the dimension of gradients due to limited channel bandwidth. It was shown that such an approach results in a much faster convergence of the learning process compared with the separate computation and communication based approaches. This work was further extended to wireless fading channels in [89].

• **Aggregation frequency control with limited bandwidth and computation resources:** The learning process includes the local updates at different devices and the global aggregation at the fusion center. We can aggregate the local updates at the interval of one or multiple local updates, such as adopting the federated averaging algorithm [19]. The aggregation frequency should be carefully designed by weighing the limited computation resources at devices locally and the limited communication bandwidth for global data aggregation. To this end, Wang *et al.* [90] provided a convergence bound of gradient-descent based federated learning from a theoretical perspective. Based on this convergence result, the authors proposed a control algorithm that learns the data distribution, system dynamics, and model characteristics, which can be used to dynamically determine the frequency of global aggregation in real time to minimize the learning loss under a fixed resource budget. Zhou and Cong [91] established the convergence results of the distributed stochastic gradient descent algorithm that is averaged after K steps for nonconvex loss functions. The convergence rate in terms of the total run time instead of the number of iterations was investigated in [92], which also proposed an adaptive communication strategy that starts with a low aggregation frequency to save communication costs, followed by increasing the aggregation frequency to achieve a low error floor.

• **Data reshuffling via index coding and pliable index coding:** Data reshuffling [202], [203] is a recognized approach to improve the statistical performance of machine learning algorithms. Randomly reshuffling the training data at each device makes the distributed learning algorithm go over the data in a different order, which brings statistical gains for non-IID data [204]. However, in edge-AI systems, its communication cost is prohibitively expensive. There are a sequence of works focusing on reducing the communication cost of data reshuffling.

To reduce the communication cost of data reshuffling, Lee *et al.* [93] proposed a coded shuffling approach based on index coding. This approach assumes that the data placement rules are pre-specified. The statistical learning performance can be improved provided a small number of new data points updated at each work, which motivates the proposal of a pliable index coding based semi-random data reshuffling approach [94] for more efficient coding schemes design. It claims that the new data for each device is not necessarily in a specific way and each data is required at no more than c devices (which is called the c-constraint). The pliable data reshuffling problem was also considered in wireless networks [95]. It was further observed that at per round it is not necessary to update a new data for all mobile devices, and the authors proposed to maximize the number of devices that are refreshed with a new data point. This approach turns out to reduce the communication cost considerably with a slight sacrifice of the learning performance.

• **Straggler mitigation via coded computing:** In practice, some devices may be stragglers during the computation of the gradients, i.e., it takes more time for these devices to finish the computation task. By carefully replicating data sets on devices, Tandon *et al.* [96] proposed to encode the computed gradients to migrate stragglers, while the amount of redundancy data depends on the number of stragglers in the system. In [97], straggler tolerance and communication cost were considered jointly. Therefore, compared with [96], the total runtime of the distributed gradient computation is further reduced by distributing the computations over subsets of gradient vector components in addition to subsets of data sets. Raviv *et al.* [98] adopted tools from classic coding theory, i.e., cyclic maximum distance separable (MDS) codes, to achieve favorable performance of gradient coding in terms of the applicable range of parameters and in the complexity of the coding algorithms. Using Reed-Solomon codes, Halbawi *et al.* [99] made the learning system more robust to stragglers compared with [96]. The performance with respect to the communication load and computation load required for mitigating the effect of stragglers was further improved in [100]. Most of straggling mitigation approaches assumed that the straggler devices have no contribution to the learning task. In contrast, it was proposed by [101] to exploit the non-persistent stragglers since they are able to complete a certain portion of assigned tasks in practice. This is achieved by transmitting multiple local updates from devices to the fusion center per communication round instead of only one local updates per round.

In addition, approximate gradient coding was proposed in [98] where the fusion center only requires an approximate computation of the full gradients instead of an exact one, which reduces the computation from the devices significantly while preserving the system tolerance to stragglers. However, this approximate gradient approach typically results in a slower convergence rate of the learning algorithm compared with the exact gradient approach [102]. When the loss function is the squared loss, it was proposed in [103] to encode the second moment of the data matrix with a low density parity-check (LDPC) code to mitigate the effect of the stragglers.

They also indicated that the moment encoding based gradient descent algorithm can be viewed as a stochastic gradient descent method, which provides opportunities to obtain convergence guarantees for the proposed approach. Considering the general loss function, it was proposed in [104] to distribute the data to the devices using low density generator matrix (LDGM) codes. Bitar *et al.* [102] proposed an approximate gradient coding scheme by distributing data points redundantly to devices based on a pair-wise balanced design, simply ignoring the stragglers. The convergence guarantees are established and the convergence rate can be improved with the redundancy of data [102].

Beyond these methods to improve communication efficiency, there are also researches [205], [206] working on optimizing wireless resources allocation during model training in distributed system modes. Abad *et al.* [205] proposed a hierarchical federated learning scheme by introducing small cell base stations orchestrating mobile devices within each cell and reducing the frequency of exchanging model updates with the aggregation center node to reduce the communication latency. Yang *et al.* [206] proposed to optimize wireless resources allocation by studying the trade-off between delay and energy consumption.

2) Decentralized System Mode: In the decentralized mode, a machine learning model is trained with a number of edge devices by exchanging information directly without a central node. A well known decentralized information exchange paradigm is the gossip communication protocol [207], by randomly evoking a node as a central node to collect updates from neighbour nodes or broadcast its local update to neighbour nodes. By integrating the gossip communication protocols into the learning algorithms, Elastic Gossip [105] and Gossiping SGD [106], [107], [108] were proposed.

One typical network topology for decentralized machine learning is the fully connected network, where each device communicates directly with all other devices. In this scenario, each device maintains a local copy of the model parameters and computes its local gradients that will be sent to all other devices. Each device can average the gradients received from every other devices and then perform local updates. In each iteration, the model parameters will be identical at all devices if each device starts from a same initial point. This process is essentially the same as the classical gradient descent at a centralized server, so the convergence can be guaranteed as in the centralized settings. However, such a fully connected network suffers a heavy communication overhead that grows quadratically in the number of devices, while the communication overhead is linear in the number of devices for centralized settings. Therefore, network topology design plays a key role in alleviating the communication bottleneck in decentralized scenarios. In addition, the convergence rate of the decentralized algorithm also depends on the topology of network [109]. We should note that the decentralized edge AI system suffers from the same issues as the system in distributed mode since each device acts like a fusion center.

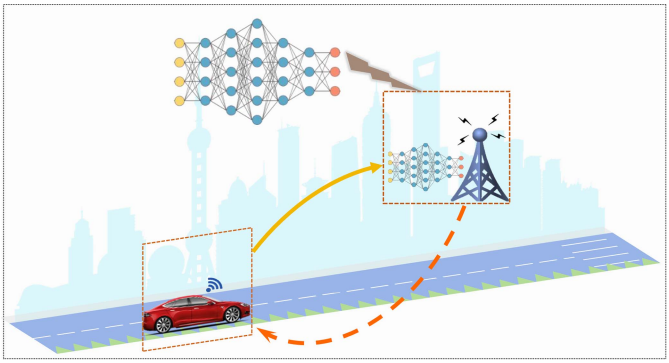
There have been several works demonstrating that some carefully designed topologies of networks achieve better performance than the fully connected network. It has been empirically observed in [110] that using an alternative network topology between devices can lead to improved learning performance in several deep reinforcement learning tasks compared with the standard fully-connected communication topology. Specifically, it was observed in [110] that the Erdos-Renyi graph topology with 1000 devices can compete with the standard fully-connected topology with 3000 devices, which shows that the machine learning performance can be more efficient if the topology is carefully designed. Considering that different devices may require different times to carry out local computation, Neglia *et al.* [111] analyzed the influences of different network topologies on the total runtime of distributed subgradient methods, which can determine the degrees of the topology graph, leading to the faster convergence speed. They also showed that a sparser network can sometimes result in significant reduction of the convergence time.

One common alternative to the fully connected network topology is to employ a ring topology [112], where each device only communicates with its neighbors that are arranged in a logical ring. More concretely, each device aggregates and passes its local gradients along the ring such that all devices have a copy of the full gradients at the end. This approach has been adopted in distributed deep learning for model updating [113], [114]. However, the algorithm deployed on the ring topology are inherently sensitive to stragglers [115]. To alleviate the effects of stragglers in the ring topology, Reisizadeh *et al.* [115] proposed to use a logical tree topology for communication, based on which they mitigated stragglers by gradient coding techniques. In the tree topology, there are several layers of devices, where each device communicates only with its parent node. By concurrently transmitting messages from a large number of children nodes to multiple parent nodes, communication with the tree topology can be more efficient than that with the ring topology.

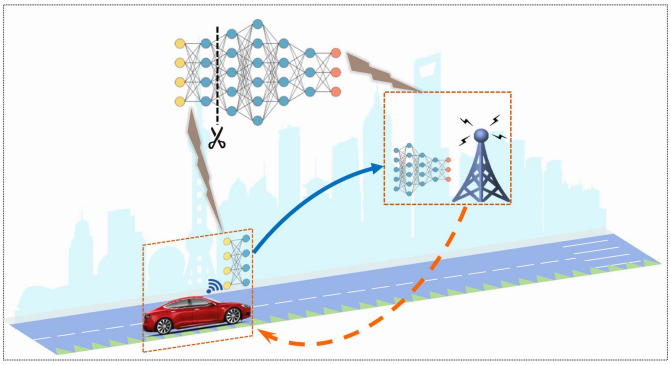
C. Model Partition Based Edge Training Systems

While data partition based edge training systems have obtained much attention in both academia and industry, there is also an important line of works designing edge AI systems based on partitioning a single machine learning model and deploying it distributedly across mobile devices and edge servers. In such systems, each node holds part of the model parameters and accomplish the model training task or the inference task collaboratively. One main advantage of model partition in the training process is the small storage size needed for each node. In this system, the machine learning model is distributedly deployed among multiple computing nodes, with each node evaluating updates of only a portion of the model's parameters. Such method is particularly useful in the scenarios where the machine learning model is too large to be stored in a single node [208], [209]. Another main concern of model partition during training is the data privacy when the data at each node belongs to different parties. However, model training with model partition based architectures also poses heavy communication overhead between edge devices.

- **Model partition across a large number of nodes to balance computation and communication:** A line of



(a) Server-based edge inference



(b) Device-edge joint inference

Fig. 6. Computation offloading based edge inference systems.

works [116], [117], [118] have considered the model partition across edge nodes with heterogeneous hardware and computing power. In [116], a reinforcement learning approach was proposed for deploying the computing graph onto edge computing devices, which, however, is time and resource intensive. To avoid the huge computation cost of reinforcement learning based approach, Narayanan *et al.* [117] proposed the PipeDream system for automatically determining the model partition strategy of DNNs. Furthermore, injecting multiple mini-batches makes the system converge faster than using a single machine or using the data partition approach. While PipeDream stresses the hardware utilization of edge devices, each device should maintain multiple versions of model parameters to avoid optimization issues caused by the staleness of parameters with asynchronous backward updates. This hinders scaling to much bigger models for PipeDream. To address this problem, the GPipe system was proposed in [118] with novel batch-splitting and re-materialization techniques, which is able to scale to large models with little additional communication overhead.

• **Model partition across the edge device and edge server to avoid the exposure of users' data:** In practice, powerful edge servers are often owned by service providers, but users may be relunctant to expose their data to service providers for model training. The observation that a DNN model can be split between two successive layers motivates researchers to deploy the first few layers on the device locally and the remaining layers on the edge server to avoid the exposure of users' data. Mao *et al.* [119] proposed a privacy-preserving deep learning architecture where the shallow part of a DNN is deployed on the mobile device and the large part is deployed on the edge server. Gupta and Raskar [210] designed a model partition approach over multiple agents, i.e., multiple data sources and one supercomputing resource, and further extended it to semi-supervised learning cases with few labeled sample. In [120], a partition approach named ARDEN was proposed by taking both privacy and performance into consideration. The model parameters at mobile device are fixed and differential privacy mechanism is introduced to guarantee the privacy of the output at mobile device. Before uploading the local output, deliberate noise is added to improve the robustness of DNN, which is shown to be beneficial for the inference performance.

• **Vertical architecture for privacy with vertically partitioned data and model:** In most industries, data is often vertically partitioned, i.e., each owner only holds partial data attributes. Data isolation becomes a severe bottleneck for collaboratively building a model due to competition, privacy, and administrative procedures. Therefore, much attention is being paid on privacy-preserving machine learning with vertically partitioned data [20]. During the training, the model is also vertically partitioned and each owner holds a part of model parameters. Therefore, *vertical architecture* of AI is proposed and studied for privacy-preserving machine learning where each node has access to different features of common data instances and maintains the corresponding subset of model parameters. What makes it worse is that the label of each data instance is only available to nodes belonging to one party.

Vaidya and Clifton [121] proposed a privacy-preserving k-means algorithm in the vertical architecture with secure multi-party computation. Kantarcioglu and Clifton [122] studied the secure association rules mining problem with vertically partitioned data. A linear regression model was taken into consideration in [123], and multi-party computation protocols were proposed with a semi-trusted third party to achieve secure and scalable training. For privacy-preserving classification with support vector machine (SVM), Yu *et al.* [124] considered the dual problem of SVM and adopted a random perturbation strategy, which is suitable only for nodes belong to more than three parties. A privacy-preserving classification approach based on decision tree was proposed in [125], which adopts secure multi-party computation procedures including commutative encryption to determine if there are any remaining attributes and secure cardinality computation of set intersection. For classification with logistic regression, the problem becomes even more difficult because of the coupled objective function as well as the gradient. To address this problem, Hardy *et al.* [126] proposed to use Taylor approximation to benefit from the homomorphic encryption protocol without revealing the data at each node.

D. Computation Offloading Based Edge Inference Systems

The advancement of edge computing makes it increasingly attractive to push the AI inference task to network edge to enable low-latency AI services for mobile users [13]. However, the power consumption and storage for DNN models is often unbearable for mobile devices such as wearable

devices. Fortunately, offloading the task from edge devices to powerful edge servers emerges as an antidote [13], [133], [134], [136]. One solution is to offload the entire inference task to an edge server, which is termed as *server-based edge inference*, as shown in Fig. 6(a). It is particularly suitable for resource limited IoT devices. In this case, the entire AI models are deployed on edge servers and edge devices should upload their input data to edge servers for inference. For latency and privacy concerns, another alternative is offloading only a part of the task to the edge server, and the edge server computes the inference result based on the intermediate value computed by the edge device. We refer to it as *device-edge joint inference* as shown in Fig. 6(b). This edge device and edge server synergy can be achieved by performing simple processing at the device and the remaining part at the edge server.

1) Server-Based Edge Inference: In the scenario where the models are deployed on the edge servers, the devices send the input data to the edge, the edge servers compute the inference results according to the trained models, and the inference results are then sent back to the devices. The main bottleneck is the limited communication bandwidth for data transmission. To reduce the real-time data transmission overhead of uplink transmission in bandwidth-limited edge AI systems, an effective way is to reduce the volume of data transmitted from devices without hurting the inference accuracy. In addition, cooperative downlink transmission of multiple edge servers has been proposed to enhance the communication efficiency for edge inference.

• **Partial data transmission:** To realize cloud based visual localization for mobile robots in real time, it is important to control the volume of the data through the network. Therefore, Ding *et al.* [127] used an data volume reduction method proposed by [128] for multi-robot communication, which employs sparsification methods to compress the data. In a cloud-based collaborative 3D mapping system, Mohanarajah *et al.* [129] proposed to reduce bandwidth requirements by sending only the key-frames as opposed to all the frames produced by the sensor, and Chen *et al.* [130] proposed to determine and offload key-frames for object detection by utilizing heuristics such as frame differences to select the key-frames. These approaches are useful in reducing the communication cost when we are able to exploit the structure of the specific tasks and the associated data.

• **Raw data encoding:** Data encoding has been widely used in compressing the data volume. For example, traditional image compression approaches (e.g., JPEG) can compress data aggressively, but they are often optimized from the perspective of human-visual, which will result in an unacceptable performance degradation in DNN applications if we use a high compression ratio. Based on this observation, to achieve a higher compression ratio, Liu *et al.* [131] proposed to optimize the data encoding schemes from the perspective of DNNs based on the frequency component analysis and rectified quantization table, which is able to achieve a higher compression ratio than the traditional JPEG method without degrading the accuracy for image recognition. Instead of using standard video encoding techniques, it was argued in [132] that data collection and transmission schemes should be designed jointly in vision tasks to maximize an end-to-end goal with a pre-trained model. Specifically, the authors proposed to use DNNs to encode the high dimensional raw data into a sparse, latent representation for efficient transmission, which can be recovered later at the cloud via a decoding DNN. In addition, this coding process is controlled by a reinforcement learning algorithm, which sends action information to devices for encoding in order to maximize the predication accuracy of the pre-trained model with decoded inputs, while achieving communication-efficient data transmission. Kurka and Gündüz [211] proposed an autoencoder based joint source-channel coding scheme for image retrieval over a wireless channel, which directly encodes the feature vector of an image as channel inputs. This novel data encoding idea is a promising solution for realizing real-time inference in edge AI systems.

• **Cooperative downlink transmission:** Cooperative transmission [212] is known as an effective aproach for improving the communication efficiency via proactive interference-aware coordination of multiple base stations. It was proposed in [133] to offload each inference task to multiple edge servers and cooperatively transmit the output results to mobile users in downlink transmission. A new technology named intelligent reflecting surface (IRS) [213] emerges as a cost-effective approach to enhance the spectrum efficiency and energy efficiency of wireless communication networks, which is promising in facilitating communication-efficient edge inference [214]. It is achieved by reconfiguring the wireless propagation environment via a planar array to induce the change of the signals' amplitude and/or phase. To further improve the performance of the cooperative edge inference scheme in [133], Hua *et al.* [134] proposed the IRS-aided edge inference system and designed a task selection strategy to minimize both the uplink and downlink transmit power consumption, as well as the computation power consumption at edge servers.

2) Device-Edge Joint Inference: For many on-device data, such as healthcare information and users' behaviors, privacy is of a primary concern. Thus, there emerges the idea of edge device and edge server synergy, which can be termed as *device-edge joint inference*, by deploying the partitioned DNN model over the mobile device and the powerful edge server. By deploying the first few layers locally, a mobile device can compute the local output with simple processing, and transmit the local output to a more powerful edge server without revealing any sensitive information.

An early work [215] considered the partition of the image classification pipeline and found that executing feature extraction on devices and offloading the rest to the edge servers achieves optimal runtime. Recently, Neurosurgeon has been proposed in [17], where a DNN model is automatically split between a device and an edge server according to the network latency for transmission and the mobile device energy consumption at different partition points, in order to minimize the total inference time. Different methods have been developed [216], [217] to partition a pre-trained DNN over several mobile devices in order to accelerate DNN inference on devices. Bhardwaj *et al.* [218] further considered memory and communication costs in this distributed inference architecture,

for which model compression and network science-based knowledge partitioning algorithm are proposed to address these issues. For robotics system where the model is partitioned between the edge server and the robot, the robot should take both local computation accuracy and offloading latency into account, and this offloading problem was formulated in [219] as a sequential decision making problem that is solved by a deep reinforcement learning algorithm.

In the following, we review the main methods for further reducing the communication overhead for the model partition based edge inference.

• **Early exit:** Early exit can be used to reduce communication workloads when partitioning DNNs, which has been proposed in [135] based on the observation that the features learned at the early layer of the network can be often sufficient to produce accurate inference results. Therefore, the inference process can exit early if the data samples can be inferred with high confidence. This technique has been adopted in [136] for distributed DNN inference over the cloud, the edge and devices. With early exit, each device first performs the first few layers of an DNN, and offloads the rest of computation to the edges or the clouds if the outputs of the device do not meet the accuracy requirements. This approach is able to reduce the communication cost by a factor of over 20× compared with the traditional approach that offloads all raw data to the cloud for inference. More recently, Li *et al.* [220] proposed an on-demand low-latency inference framework through jointly designing the model partition strategy according to the heterogeneous computation capabilities between a mobile device and edge servers, and the early exit strategy according to the complicated network environment.

• **Encoded transmission and pruning for compressing the transmitted data:** In a hierarchical distributed architecture, the main communication bottleneck is that the transmission of intermediate values between the partition point since the intermediate data can be much larger than the raw data. To reduce the communication overhead of intermediate value transmissions, it was proposed in [137] to partition a network at an intermediate layer, whose features are encoded before wireless transmissions to reduce their data size. It shows that partitioning a CNN at the end of the last convolutional layer where the data communication requirement is less coupled with feature space encoding enables significant reduction in communication workloads. Recently, a deep learning based end-to-end architecture was proposed in [221], named BottleNet++. By jointly considering model partition, feature compression and transmission, BottleNet++ achieves up to 64x bandwidth reduction over the additive white Gaussian noise channel and up to 256x bit compression ratio in the binary erasure channel, with less than 2% reduction in accuracy, compared with merely transmitting intermediate data without feature compression.

Network pruning, as discussed in Section III-D2, has been exploited in reducing the communication overhead of intermediate feature transmissions. For example, a 2-step pruning approach was proposed in [138] to reduce the transmission workload at the network partition point by limiting the pruning region. Specifically, the first step aims to reduce the total computation workload of the network while the second step aims to compress the intermediate data for transmission.

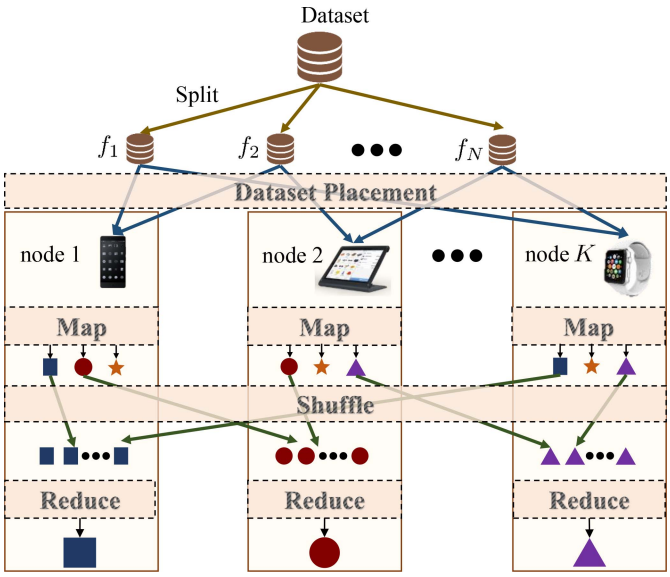


Fig. 7. MapReduce computation model [31].

• **Coded computing for cooperative edge inference:** Coding theory can be leveraged to address the communication challenges of distributed inference in edge AI systems. For example, Zhang and Simeone [139] considered distributed linear inference in mobile edge AI system, where the model is partitioned among several edge devices that compute the inference results cooperatively for each device. It was shown in [139] that coding is efficient in reducing the overall computation-plus-communication latency.

E. General Edge Computing System

Beyond the edge AI system architectures mentioned above, there are also edge AI systems based on a general computing paradigm, namely, MapReduce. MapReduce [193] is a general distributed computing framework that is able to achieve parallel speedups on a variety of machine learning problems during training and inference procedures [222]. The MapReduce-like distributed computing framework takes the distributed data input and distributed model deployment into account jointly. In [223], the convolutional neural network was implemented based on the MapReduce framework to accelerate its training process. Ghoting *et al.* [224] proposed SystemML based on the MapReduce framework to support distributed training for a broad class of supervised and unsupervised machine learning algorithms. Yang *et al.* [31] proposed a communication-efficient wireless data shuffling strategy for supporting MapReduce-based distributed inference tasks.

In the MapReduce-like distributed computing framework as shown in Fig. 7, there are generally three phases (i.e., a map phase, a shuffle phase, and a reduce phase) to complete a computational task. In the map phase, every computing node computes a map function of the assigned data simultaneously, generating a number of intermediate values. In

the shuffle phase, nodes communicate with each other to obtain some intermediate values for computing the output function. Subsequently, in the reduce phase, each node computes the assigned output function according to the available intermediate values. However, there are two main bottlenecks in such a distributed computing framework. One is the heavy communication load in the shuffle phase, and another is the straggler delay caused by the variability of computation times at different nodes. To address these problems, coding has been proposed as a promising approach by exploiting abundant computing resources at the network edge [140]. In recent years, coding techniques are becoming a hot area of research for reducing the communication cost of data shuffling, as well as reducing the computing latency by mitigating straggler nodes, as reviewed below.

• **Coding techniques for efficient data shuffling:** Coding techniques for shuffling data in the MapReduce-like distributed computing framework were first proposed in [141], which considered a wireline scenario where each computing node can obtain the intermediate values from other nodes through a shared link. In [30], the authors extended the work in [141] to a wireless setting, where the computing nodes are able to communicate with each other via an access point. A scalable data shuffling scheme was proposed by utilizing a particular repetitive pattern of placing intermediate values among devices, reducing the communication bandwidth by factors that grow linearly with the number of devices. To improve the wireless communication efficiency (i.e., achieved data rates) in the data shuffle phase, a low-rank optimization model was proposed in [31] by establishing the interference alignment condition. The low-rank model is further solved by an efficient difference-of-convex-functions (DC) algorithm. Both [30] and [31] considered the communication load minimization problem under the wireless communication setting with a central node.

There are also some works considering the problem of reducing the communication load in data shuffling under the wireless communication scenario without a coordinating center. That is, the computing nodes can communicate with each other through an shared wireless interference channel. For example, assuming perfect channel state information, a beamforming strategy was proposed in [142] based on side information cancellation and zero-forcing to trade the abundant computing nodes for reducing communication load, which outperforms the coded TDMA broadcast scheme based on [141]. This work was further extended in [143] to consider imperfect channel state information. The paper [144] proposed a dataset cache strategy and a coded transmission strategy for the corresponding computing results. The goal is to minimize the communication load characterized by latency (in seconds) instead of channel uses (in bits), which is more practical in wireless networks. In [145], the authors noted that to trade abundant computation for the communication load, the computational tasks must be divided into an extremely large number of subtasks, which is impractical. Therefore, they proposed to ameliorate this limitation by node cooperation and designed an efficient scheme for task assignment. Prakash *et al.* [146] investigated coded computing for distributed graph processing systems, which improves performance significantly compared with the general MapReduce framework by leveraging the structure of graphs.

• **Coding techniques for straggler mitigation:** Another line of work focuses on addressing the straggler problem in distributed computing by coding techniques. Mitigating the effect of stragglers utilizing coding theory was first proposed in [145] for a wired network. The main idea is to leverage redundant computing nodes to perform computational subtasks, while the computation result can be correctly recovered as long as the local computation results from any desired subset of computing nodes are collected. This work was extended to wireless networks [147], where only one local computing node can send its computation results to the fusion center at a time. The paper [148] proposed a sub-task assignment method to minimize the total latency which is composed of the latency caused by wireless communication between different computing nodes and the fusion center and the latency caused by the variability of computation time of different devices. Most of the above works focused on linear computations (e.g., matrix multiplication). However, to realize the distributed inference on state-of-the-art machine learning algorithms (e.g., DNN), non-linear computation should be taken into consideration. As a result, the work [149] proposed a learning-based approach to design codes that can handle the stragglers issue in distributed non-linear computation problems. Ozfatura *et al.* [225] studied a multi-message communication problem and designed straggler avoidance techniques, which enjoy advantages over methods based on transmitting only one message for a computation node per iteration to the center node.

From the discussions of communication-efficient edge AI systems in this section, we find that the availability of source data and the structure of AI tasks are two main influences on determining the architecture of edge AI systems. Based on different system architectures, there are many efforts to fit the constraints (e.g., storage, computation, communication, privacy, etc.) in various applications, yielding an important and active research area.

V. Conclusion and Future Directions

This paper presented a comprehensive survey on the communication challenges and solutions in edge AI systems, which shall support a plethora of AI-enabled applications at the network edge. Specifically, we first summarized communication efficient algorithms for distributed training AI models on edge nodes, including zeroth-order, first-order, second-order, and federated optimization algorithms. We then categorized different system architectures of edge AI systems, including data partition based, and model partition based edge training systems. Next, we revisited works bridging the gap between computation offloading and edge inference. Beyond these system architectures, we also introduced general edge computing defined AI systems. The communication issues and solutions in such architectures were extensively discussed. As we have discussed from the algorithm level and system level

throughout the paper, it is usually infeasible to find a unified framework for various edge AI applications. It has great potential to study the joint design of edge AI system and algorithm according to the type of AI tasks and the characteristics of each involved node. The most important lesson we have learned is probably that determining "what to transmit" is often much beneficial for reducing the communication overhead of an edge AI system. This perspective of learning has motivated a line of works to further improve the communication efficiency compared with optimizing "how to transmit" from the perspective of traditional communication theory.

The activities and applications of edge AI are growing rapidly. Note that it is also critical to develop hardware and software to facilitate the implementations and applications of edge AI, for which there are still a number of challenges and future directions as listed below.

- **Edge AI hardware design:** Hardware of edge nodes determines the physical limits of AI systems, for which there are a growing amount of efforts on edge AI hardware design. For example, Google edge tensor processing unit (TPU) is designed for high-speed inference at the edge. Nvidia has rolled out Jetson TX2 for power-efficient embedded AI computing. Nevertheless, these hardwares mainly focus on performing the entire task, especially edge inference locally. In the future, a variety of edge AI hardwares will be customized for different AI system architectures and applications.
- **Edge AI software platforms:** The past decade has witnessed the blossom of AI software platforms from top companies for supporting cloud-based AI services. Cloud-based AI service providers are trying to include edge nodes into their platforms, though edge nodes only serve as simple extensions of cloud computing nodes currently. Google Cloud IoT, Microsoft Azure IoT, NVIDIA EGX, and Amazon Web Services (AWS) IoT are able to connect IoT devices to cloud platforms, thereby managing edge devices and processing the data from all kinds of IoT devices.
- **Edge AI as a service:** For different fields and applications [226], there are a variety of additional design targets and constraints, thereby requiring domain-specific edge AI frameworks. Edge AI will be a service infrastructure that integrates the computation, communication, storage and power resources at network edges to enable data-driven intelligent applications. A notable example in credit industry is FATE [227], an industrial grade federated learning framework proposed by Webank. A number of federated learning algorithms [192], [228] were designed to break data isolation among institutions and to preserve the data privacy during edge training. Another representative attempt of edge AI for smart healthcare is NVIDIA Clara [229], which delivers AI to healthcare and life sciences with NVIDIA's EGX edge computing platform. Since Clara features federated learning, it supports an innovative approach to collaboratively build a healthcare AI model from hospitals and medical institutions, while protecting patient data.

ACKNOWLEDGMENT

The authors sincerely thank Prof. Zhi Ding from the University of California at Davis for insightful and constructive comments to improve the presentation of this work.

REFERENCES

[1] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2013, pp. 6645–6649.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2012, pp. 1097–1105.
[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
[4] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of Vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.
[5] S. Haddadin, L. Johannsmeier, and F. D. Ledezma, "Tactile robots as a central embodiment of the tactile Internet," *Proc. IEEE*, vol. 107, no. 2, pp. 471–487, Feb. 2019.
[6] J. Bughin and J. Seong, *Assessing the Economic Impact of Artificial Intelligence*, ITU, Geneva, Switzerland, Sep. 2018.
[7] *Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper*, Cisco, San Jose, CA, USA, Nov. 2018.
[8] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Nov. 2017.
[9] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.
[10] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.
[11] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, Aug. 2019.
[12] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, 2015.
[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[14] A. A. Al-Habob and O. A. Dobre. (Nov. 2019). *Mobile Edge Computing and Artificial Intelligence: A Mutually-Beneficial Relationship*. [Online] Available: https://www.comsoc.org/publications/tcn/2019-nov/mobile-edge-computing-and-artificial-intelligence-mutually-beneficial-relationship.
[15] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Towards an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, Jan. 2020.
[16] J. Konecny, H. B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," in *Proc. NIPS Optim. Mach. Learn. Workshop*, 2015, p. 5.
[17] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, pp. 615–629, Apr. 2017.
[18] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2017, pp. 1273–1282.
[20] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12, 2019.
[21] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Commun. Mag.*, vol. 44, no. 4, pp. 98–107, Apr. 2006.
[22] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 2022–2035, Jan. 2020.

[23] D. J. Love, R. W. Heath, V. K. Lau, D. Gesbert, B. D. Rao, and M. Andrews, "An overview of limited feedback in wireless communication systems," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 8, pp. 1341–1365, Oct. 2008.

[24] Y. Du, S. Yang, and K. Huang, "High-dimensional stochastic gradient quantization for communication-efficient edge learning," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, 2019, pp. 1–5.

[25] S. Reddy, J. Fox, and M. P. Purohit, "Artificial intelligence-enabled healthcare delivery," *J. Royal Soc. Med.*, vol. 112, no. 1, pp. 22–28, 2019.

[26] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun, "Using recurrent neural network models for early detection of heart failure onset," *J. Amer. Med. Inf. Assoc.*, vol. 24, no. 2, pp. 361–370, 2016.

[27] O. Gottesman *et al.*, "Guidelines for reinforcement learning in healthcare," *Nat. Med.*, vol. 25, no. 1, pp. 16–18, 2019.

[28] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1135–1143.

[29] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.

[30] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2643–2654, Oct. 2017.

[31] K. Yang, Y. Shi, and Z. Ding, "Data shuffling in wireless distributed computing via low-rank optimization," *IEEE Trans. Signal Process.*, vol. 67, no. 12, pp. 3087–3099, Jun. 2019.

[32] P. Sun, W. Feng, R. Han, S. Yan, and Y. Wen, "Optimizing network performance for distributed DNN training on GPU clusters: ImageNet/AlexNet training in 1.5 minutes," 2019. [Online]. Available: arXiv:1902.06855.

[33] EU, "Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (general data protection regulation)," *Official J. Eur. Union*, vol. 119, pp. 1–88, Apr. 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L:2016:119:FULL&from=EN

[34] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *ACM Meas. Anal. Comput. Syst.*, vol. 1, pp. 1–25, Dec. 2017.

[35] Y. Dong, J. Cheng, M. J. Hossain, and V. C. M. Leung, "Secure distributed on-device learning networks with Byzantine adversaries," *IEEE Netw.*, vol. 33, no. 6, pp. 180–187, Nov. 2019.

[36] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[37] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[38] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[39] K. Yuan, B. Ying, J. Liu, and A. H. Sayed, "Variance-reduced stochastic learning by networked agents under random reshuffling," *IEEE Trans. Signal Process.*, vol. 67, no. 2, pp. 351–366, Jan. 2019.

[40] J. D. Lee, Q. Lin, T. Ma, and T. Yang, "Distributed stochastic variance reduced gradient methods by sampling extra data with replacement," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 4404–4446, 2017.

[41] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–8.

[42] M. Murshed *et al.*, "Machine learning at the network edge: A survey," 2019. [Online]. Available: arXiv:1908.00080.

[43] S. Deng, H. Zhao, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," 2019. [Online]. Available: arXiv:1909.00560.

[44] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 2, pp. 253–279, May 2019.

[45] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: The power of two function evaluations," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2788–2806, May 2015.

[46] D. Yuan and D. W. C. Ho, "Randomized gradient-free method for multiagent optimization over time-varying networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1342–1347, Jun. 2015.

[47] A. K. Sahu, D. Jakovetic, D. Bajovic, and S. Kar, "Distributed zeroth order optimization over random networks: A Kiefer–Wolfowitz stochastic approximation approach," in *Proc. IEEE Conf. Decis. Control (CDC)*, 2018, pp. 4951–4958.

[48] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 5050–5060.

[49] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar, "Communication-efficient distributed reinforcement learning," 2018. [Online]. Available: arXiv:1812.03239.

[50] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, "AdaComp: Adaptive residual gradient compression for data-parallel distributed training," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2827–2835.

[51] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-Newton method for online convex optimization," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2007, pp. 436–443.

[52] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-Newton method for large-scale optimization," *SIAM J. Optim.*, vol. 26, no. 2, pp. 1008–1031, 2016.

[53] P. Moritz, R. Nishihara, and M. Jordan, "A linearly-convergent stochastic L-BFGS algorithm," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2016, pp. 249–258.

[54] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate Newton-type method," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 1000–1008.

[55] Y. Zhang and X. Lin, "DiSCO: Distributed optimization for self-concordant empirical loss," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 362–370.

[56] S. Wang, F. Roosta-Khorasani, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate Newton method for distributed optimization," in *Proc. NeurISP*, 2018, pp. 2332–2342.

[57] C. Dünner, A. Lucchi, M. Gargiani, A. Bian, T. Hofmann, and M. Jaggi, "A distributed second-order algorithm you can trust," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1358–1366.

[58] M. Jaggi *et al.*, "Communication-efficient distributed dual coordinate ascent," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2014, pp. 3068–3076.

[59] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. ICML Workshop Adapt. Multitask Learn.*, 2020, pp. 424–436.

[60] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 4427–4437.

[61] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014. [Online]. Available: arXiv:1412.6115.

[62] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2015, pp. 3123–3131.

[63] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016. [Online]. Available: arXiv:1602.02830.

[64] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNoR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 525–542.

[65] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 2285–2294.

[66] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing convolutional neural networks in the frequency domain," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2016, pp. 1475–1484.

[67] Y. Lin, Z. Song, and L. F. Yang, "Towards a theoretical understanding of hashing-based neural nets," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2019, pp. 751–760.

[68] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 1990, pp. 598–605.

[69] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 1993, pp. 164–171.

[70] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2015, pp. 1135–1143.

[71] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 3123–3131.

[72] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Proc. Adv. Neural. Inf. Process. Syst. (NIPS)*, 2017, pp. 3288–3298.

[73] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-trim: Convex pruning of deep neural networks with performance guarantee," in *Proc. Adv. Neural. Inf. Process. Syst. (NIPS)*, 2017, pp. 3180–3189.

[74] T. Jiang, X. Yang, Y. Shi, and H. Wang, "Layer-wise deep neural network pruning via iteratively reweighted optimization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2019, pp. 5606–5610.

[75] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2016, pp. 2554–2564.

[76] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural. Inf. Process. Syst. (NIPS)*, 2016, pp. 2074–2082.

[77] S. Oymak, "Learning compact neural networks with regularization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 3963–3972.

[78] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2013, pp. 6655–6659.

[79] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2014, pp. 1269–1277.

[80] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proc. British Mach. Vision Conf. (BMVC)*, 2014.

[81] M. Denil, B. Shakibi, L. Dinh, M. A. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural. Inf. Process. Syst. (NIPS)*, 2013, pp. 2148–2156.

[82] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 2754–2761.

[83] S. Amos, T. Bugra, R. Roberto, L. Vincent, and F. Pascal, "Learning separable filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 94–106, Jan. 2015.

[84] V. Sindhwani, T. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2015, pp. 3088–3096.

[85] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 2857–2865.

[86] Z. Yang *et al.*, "Deep fried ConvNets," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1476–1483.

[87] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[88] M. M. Amiri and D. Gunduz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[89] M. M. Amiri and D. Gunduz, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020.

[90] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[91] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 3219–3227.

[92] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," 2018. [Online]. Available: arXiv:1810.08313.

[93] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[94] L. Song, C. Fragouli, and T. Zhao, "A pliable index coding approach to data shuffling," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2558–2562.

[95] T. Jiang, K. Yang, and Y. Shi, "Pliable data shuffling for on-device distributed learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2019, pp. 7460–7464.

[96] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 70, 2017, pp. 3368–3376.

[97] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5606–5615.

[98] N. Raviv, R. Tandon, A. Dimakis, and I. Tamo, "Gradient coding from cyclic MDS codes and expander graphs," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4302–4310.

[99] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed–Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 2027–2031.

[100] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, 2018, pp. 857–866.

[101] E. Ozfaturay, D. Gunduz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Jul. 2019, pp. 2729–2733.

[102] R. Bitar, M. Wootters, and S. E. Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," 2019. [Online]. Available: arXiv:1905.05383.

[103] R. K. Maity, A. S. Rawat, and A. Mazumdar, "Robust gradient descent via moment encoding and LDPC codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Jul. 2019, pp. 2734–2738.

[104] R. K. Maity, A. S. Rawat, and A. Mazumdar, "Distributed stochastic gradient descent using LDGM codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1417–1421.

[105] S. Pramod, "Elastic gossip: Distributing neural network training using gossip-like protocols," 2018. [Online]. Available: arXiv:1812.02407.

[106] J. Daily, A. Vishnu, C. Siegel, T. Warfel, and V. Amatya, "GossipGrad: Scalable deep learning using gossip communication based asynchronous gradient descent," 2018. [Online]. Available: arXiv:1803.05880.

[107] M. Blot, D. Picard, N. Thome, and M. Cord, "Distributed optimization for deep learning with gossip exchange," *Neurocomputing*, vol. 330, pp. 287–296, Feb. 2019.

[108] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," in *Proc. NIPS Optim. Mach. Learn. Workshop*, 2016, pp. 482–491.

[109] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proc. IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[110] D. Adjodah *et al.*, "Communication topologies between learning agents in deep reinforcement learning," 2019. [Online]. Available: arXiv:1902.06740.

[111] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *Proc. INFOCOM*, 2019, pp. 2350–2358.

[112] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *J. Parallel Distrib. Comput.*, vol. 69, no. 2, pp. 117–124, 2009.

[113] P. H. Jin, Q. Yuan, F. Iandola, and K. Keutzer, "How to scale distributed deep learning?" 2016. [Online]. Available: arXiv:1611.04581.

[114] A. Sergeev and M. Del Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," 2018. [Online]. Available: arXiv:1802.05799.

[115] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "CodedReduce: A fast and robust framework for gradient aggregation in distributed learning," 2019. [Online]. Available: arXiv:1902.01981.

[116] A. Mirhoseini *et al.*, "Device placement optimization with reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2430–2439.

[117] D. Narayanan *et al.*, "PipeDream: Generalized pipeline parallelism for DNN training," in *Proc. ACM Symp. Oper. Syst. Principles*, 2019, pp. 1–15.

[118] Y. Huang *et al.*, "GPipe: Efficient training of giant neural networks using pipeline parallelism," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 103–112.

[119] Y. Mao, S. Yi, Q. Li, J. Feng, F. Xu, and S. Zhong, "A privacy-preserving deep learning approach for face recognition with edge computing," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 308–318.

[120] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2018, pp. 2407–2416.

[121] J. Vaidya and C. Clifton, "Privacy-preserving *k*-means clustering over vertically partitioned data," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2003, pp. 206–215.

[122] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1026–1037, Sep. 2004.

[123] A. Gascón *et al.*, "Secure linear regression on vertically partitioned datasets," in *Proc. IACR Cryptol. ePrint Archive*, vol. 2016, 2016, p. 892.

[124] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving SVM classification on vertically partitioned data," in *Proc. Pac.–Asia Conf. Knowl. Disc. Data Min.*, 2006, pp. 647–656.

[125] J. Vaidya and C. Clifton, "Privacy-preserving decision trees over vertically partitioned data," in *Proc. IFIP Annu. Conf. Data Appl. Security Privacy*, 2005, pp. 139–152.

[126] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017. [Online]. Available: arXiv:1711.10677.

[127] X. Ding, Y. Wang, L. Tang, H. Yin, and R. Xiong, "Communication constrained cloud-based long-term visual localization in real time," 2019. [Online]. Available: arXiv:1903.03968.

[128] L. Paull, G. Huang, M. Seto, and J. J. Leonard, "Communication-constrained multi-AUV cooperative slam," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 509–516.

[129] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based collaborative 3D mapping in real-time with low-cost robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 423–431, Apr. 2015.

[130] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "GLIMPSE: Continuous, real-time object recognition on mobile devices," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2015, pp. 155–168.

[131] Z. Liu *et al.*, "DeepN-JPEG: A deep neural network favorable jpeg-based image compression framework," in *Proc. Annu. Design Autom. Conf.*, 2018, p. 18.

[132] S. P. Chinchali, E. Cidon, E. Pergament, T. Chu, and S. Katti, "Neural networks meet physical networks: Distributed inference between edge devices and the cloud," in *Proc. ACM Workshop Hot Topics Netw.*, 2018, pp. 50–56.

[133] K. Yang, Y. Shi, W. Yu, and Z. Ding, "Energy-efficient processing and robust wireless cooperative transmission for edge inference," 2019. [Online]. Available: arXiv:1907.12475.

[134] S. Hua, Y. Zhou, K. Yang, and Y. Shi, "Reconfigurable intelligent surface for green edge inference," 2019. [Online]. Available: arXiv:1912.00820.

[135] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. Int. Conf. Pattern Recognit. (ICPR)*, 2016, pp. 2464–2469.

[136] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 328–339.

[137] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveillance (AVSS)*, 2018, pp. 1–6.

[138] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," 2019. [Online]. Available: arXiv:1903.03472.

[139] J. Zhang and O. Simeone, "On model coding for distributed inference and transmission in mobile edge computing systems," *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 1065–1068, Jun. 2019.

[140] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, Apr. 2017.

[141] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.

[142] F. Li, J. Chen, and Z. Wang, "Wireless MapReduce distributed computing," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2018, pp. 1286–1290.

[143] S. Ha, J. Zhang, O. Simeone, and J. Kang, "Wireless Map-Reduce distributed computing with full-duplex radios and imperfect CSI," in *Proc. SPAWC*, 2019, pp. 1–5.

[144] M. Ji and R.-R. Chen, "Fundamental limits of wireless distributed computing networks," in *Proc. INFOCOM*, 2018, pp. 2600–2608.

[145] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1291–1295.

[146] S. Prakash, A. Reisizadeh, R. Pedarsani, and S. Avestimehr, "Coded computing for distributed graph analytics," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1221–1225.

[147] A. Reisizadeh and R. Pedarsani, "Latency analysis of coded computation schemes over wireless networks," in *Proc. 55th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2017, pp. 1256–1263.

[148] S. Zhao, "A node-selection-based sub-task assignment method for coded edge computing," *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 797–801, May 2019.

[149] J. Kosaian, K. Rashmi, and S. Venkataraman, "Learning a code: Machine learning for approximate non-linear coded computation," 2018. [Online]. Available: arXiv:1806.01259.

[150] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Found. Comput. Math.*, vol. 17, no. 2, pp. 527–566, 2017.

[151] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. ACM Workshop Artif. Intell. Security*, 2017, pp. 15–26.

[152] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, vol. 8. Philadelphia, PA, USA: SIAM, 2009.

[153] D. Yuan, D. W. Ho, and S. Xu, "Zeroth-order method for distributed optimization with approximate projections," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 284–294, Feb. 2015.

[154] Y. Pang and G. Hu, "Exact convergence of gradient-free distributed optimization method in a multi-agent system," in *Proc. IEEE Conf. Decis. Control (CDC)*, 2018, pp. 5728–5733.

[155] D. Hajinezhad, M. Hong, and A. Garcia, "ZONE: Zeroth order non-convex multi-agent optimization over networks," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 3995–4010, Oct. 2019.

[156] A. K. Sahu, D. Jakovetic, D. Bajovic, and S. Kar, "Non-asymptotic rates for communication efficient distributed zeroth order strongly convex optimization," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, 2018, pp. 628–632.

[157] A. K. Sahu, D. Jakovetic, D. Bajovic, and S. Kar, "Communication-efficient distributed strongly convex stochastic optimization: Non-asymptotic rates," 2018. [Online]. Available: arXiv:1809.02920.

[158] J. Ding, D. Yuan, G. Jiang, and Y. Zhou, "Distributed quantized gradient-free algorithm for multi-agent convex optimization," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, 2017, pp. 6431–6435.

[159] X. Chen, J. Wang, and H. Ge, "signSGD via zeroth-order oracle," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, p. 6.

[160] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2010, pp. 2595–2603.

[161] Y. Zhang, J. C. Duchi, and M. J. Wainwright, "Communication-efficient algorithms for statistical optimization," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 3321–3363, 2013.

[162] O. Shamir and N. Srebro, "Distributed stochastic optimization and learning," in *Proc. 52nd Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2014, pp. 850–857.

[163] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–6.

[164] D. Yin, A. Pananjady, M. Lam, D. Papailiopoulos, K. Ramchandran, and P. Bartlett, "Gradient diversity: A key ingredient for scalable distributed learning," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2018, pp. 1998–2007.

[165] P. Goyal *et al.*, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," 2017. [Online]. Available: arXiv:1706.02677.

[166] W. Zhang *et al.*, "Distributed deep learning strategies for automatic speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2019, pp. 5706–5710.

[167] Y. Arjevani and O. Shamir, "Communication complexity of distributed convex learning and optimization," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2015, pp. 1756–1764.

[168] Y. Zhang and L. Xiao, "Communication-efficient distributed optimization of self-concordant empirical loss," in *Large-Scale and Distributed Optimization*. Cham, Switzerland: Springer, 2018, pp. 289–341.

[169] D. Garber, O. Shamir, and N. Srebro, "Communication-efficient algorithms for distributed stochastic principal component analysis," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1203–1212.

[170] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3329–3337.

[171] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 1709–1720.

[172] S. M. M. Kalan, M. Soltanolkotabi, and S. Avestimehr, "Fitting ReLUs via SGD and quantized SGD," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2019, pp. 2469–2473.

[173] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1058–1062.

[174] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SIGNSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 559–568.

[175] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 7652–7662.

[176] M. Yu *et al.*, "GradiVeQ: Vector quantization for bandwidth-efficient gradient aggregation in distributed CNN training," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 5129–5139.

[177] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, vol. 159. New York, NY, USA: Springer, 2012.

[178] J. Jiang, F. Fu, T. Yang, and B. Cui, "SketchML: Accelerating distributed machine learning with data sketches," in *Proc. ACM Int. Conf. Manag. Data*, 2018, pp. 1269–1284.

[179] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1488–1492.

[180] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 1299–1309.

[181] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 5973–5983.

[182] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, nos. 1–3, pp. 503–528, 1989.

[183] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016. [Online]. Available: arXiv:1610.05492.

[184] Y. Zhang, M. J. Wainwright, and J. C. Duchi, "Communication-efficient algorithms for statistical optimization," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2012, pp. 1502–1510.

[185] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," 2018. [Online]. Available: arXiv:1808.07576.

[186] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1387–1395.

[187] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," 2019. [Online]. Available: arXiv:1909.07972.

[188] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*, 2016, pp. 4820–4828.

[189] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1737–1746.

[190] M. Pilanci and M. J. Wainwright, "Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1842–1879, 2016.

[191] H. Choi, T. Jiang, and Y. Shi, "Large-scale beamforming for massive MIMO via randomized sketching," 2019. [Online]. Available: arXiv:1903.05904.

[192] K. Yang, T. Fan, T. Chen, Y. Shi, and Q. Yang, "A quasi-Newton method based vertical federated learning framework for logistic regression," in *Proc. NeurIPS Workshops Feder. Learn. Data Privacy Confidentiality*, 2019.

[193] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[194] B. Nazer and M. Gastpar, "Computation over multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3498–3516, Oct. 2007.

[195] M. Goldenbaum, H. Boche, and S. Stańczak, "Harnessing interference for analog function computation in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 61, no. 20, pp. 4893–4906, Oct. 2013.

[196] G. Zhu and K. Huang, "MIMO over-the-air computation for high-mobility multi-modal sensing," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6089–6103, Aug. 2019.

[197] J. Dong, Y. Shi, and Z. Ding, "Blind over-the-air computation and data fusion via provable Wirtinger flow," *IEEE Trans. Signal Process.*, vol. 68, pp. 1136–1151, Mar. 2020.

[198] F. Wu, L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "NOMA-enhanced computation over multi-access channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2252–2267, Apr. 2020.

[199] W.-T. Chang and R. Tandon, "Communication efficient federated learning over multiple access channels," 2020. [Online]. Available: arXiv:2001.08737.

[200] Y. Dong, "Distributed sensing with orthogonal multiple access: To code or not to code?" *IEEE Trans. Signal Process.*, vol. 68, pp. 1315–1330, Feb. 2020.

[201] G. Zhu, Y. Du, D. Gunduz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," 2020. [Online]. Available: arXiv:2001.05713.

[202] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Math. Program. Comput.*, vol. 5, no. 2, pp. 201–226, 2013.

[203] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, "Why random reshuffling beats stochastic gradient descent," *Math. Program.*, pp. 1–36, Oct. 2019.

[204] M. A. Attia and R. Tandon, "Near optimal coded data shuffling for distributed learning," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 7325–7349, Nov. 2019.

[205] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2020, pp. 8866–8870.

[206] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," 2019. [Online]. Available: arXiv:1911.02417.

[207] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 2508–2530, Jun. 2006.

[208] R. Mayer and H.-A. Jacobsen, "Scalable deep learning on distributed infrastructures: Challenges, techniques and tools," 2019. [Online]. Available: arXiv:1903.11314.

[209] E. P. Xing *et al.*, "Petuum: A new platform for distributed machine learning on big data," *IEEE Trans. Big Data*, vol. 1, no. 2, pp. 49–67, Sep. 2015.

[210] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *J. Netw. Comput. Appl.*, vol. 116, pp. 1–8, Oct. 2018.

[211] D. B. Kurka and D. Gündüz, "Deep joint source-channel coding of images with feedback," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2020, pp. 5235–5239.

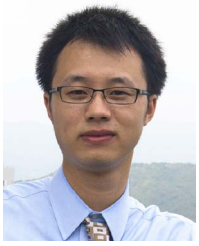
[212] D. Gesbert, S. Hanly, H. Huang, S. S. Shitz, O. Simeone, and W. Yu, "Multi-cell MIMO cooperative networks: A new look at interference," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 9, pp. 1380–1408, Dec. 2010.

[213] W. Qingqing and Z. Rui, "Towards smart and reconfigurable environment: Intelligent reflecting surface aided wireless network," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 106–112, Jan. 2020.

[214] X. Yuan, Y.-J. Zhang, Y. Shi, W. Yan, and H. Liu, "Reconfigurable-intelligent-surface empowered 6G wireless communications: Challenges and opportunities," 2020. [Online]. Available: arXiv:2001.00364.

[215] J. Hauswald, T. Manville, Q. Zheng, R. Dreslinski, C. Chakrabarti, and T. Mudge, "A hybrid approach to offloading mobile image classification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2014, pp. 8375–8379.

[216] J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, "MoDNN: Local distributed mobile computing system for deep neural network," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2017, pp. 1396–1401.
[217] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.
[218] K. Bhardwaj, C. Lin, A. Sartor, and R. Marculescu, "Memory-and communication-aware model compression for distributed deep learning inference on IoT," 2019. [Online]. Available: arXiv:1907.11804.
[219] S. Chinchali *et al.*, "Network offloading policies for cloud robotics: A learning-based approach," 2019. [Online]. Available: arXiv:1902.05703.
[220] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.
[221] J. Shao and J. Zhang, "BottleNet++: An end-to-end approach for feature compression in device-edge co-inference systems," in *Proc. IEEE Int. Conf. Commun. (ICC) Workshop Edge Mach. Learn. 5G Mobile Netw. Beyond*, Dublin, Ireland, Jun. 2019.
[222] C.-T. Chu *et al.*, "Map-reduce for machine learning on multicore," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2007, pp. 281–288.
[223] N. Basit *et al.*, "MapReduce-based deep learning with handwritten digit recognition case study," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2016, pp. 1690–1699.
[224] A. Ghoting *et al.*, "SystemML: Declarative machine learning on MapReduce," in *Proc. Int. Conf. Data Eng.*, 2011, pp. 231–242.
[225] E. Ozfatura, S. Ulukus, and D. Gündüz, "Straggler-aware distributed learning: Communication–computation latency trade-off," *Entropy*, vol. 22, no. 5, p. 544, 2020.
[226] J. C. Guevara, R. D. S. Torres, and N. L. da Fonseca, "On the classification of fog computing applications: A machine learning perspective," *J. Netw. Comput. Appl.*, vol. 159, Jun. 2020, Art. no. 102596.
[227] WeBank. (2018). *FATE: An Industrial Grade Federated Learning Framework*. [Online]. Available: https://fate.fedai.org
[228] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "SecureBoost: A lossless federated learning framework," 2019. [Online]. Available: arXiv:1901.08755.
[229] NVIDIA. (2019). *NVIDIA Clara: An Application Framework Optimized for Healthcare and Life Sciences Developers*. [Online]. Available: https://developer.nvidia.com/clara



Yuanming Shi (Member, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2011, and the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, in 2015. Since September 2015, he has been with the School of Information Science and Technology, ShanghaiTech University, where he is currently a tenured Associate Professor. He visited the University of California, Berkeley, CA, USA, from October 2016 to February 2017. His research areas include optimization, statistics, machine learning, signal processing, and their applications to 6G, IoT, and AI. He is a recipient of the 2016 IEEE Marconi Prize Paper Award in Wireless Communications, and the 2016 Young Author Best Paper Award by the IEEE Signal Processing Society. He is an Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.

Kai Yang (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from the Dalian University of Technology, China, in 2015, and the Ph.D. degree in communication and information system from the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China, in 2020. His research focuses on designing efficient systems and optimization algorithms for wireless communication, distributed computing, machine learning, and federated learning in particular.

Tao Jiang (Graduate Student Member, IEEE) received the B.S. degree in communication engineering from Xidian University, Xi'an, China, in 2017, and the M.S. degree in computer science from ShanghaiTech University, Shanghai, China, in 2020. His main research interests include optimization theory, machine learning, and wireless communications.

Jun Zhang (Senior Member, IEEE) received the B.Eng. degree in electronic engineering from the University of Science and Technology of China in 2004, the M.Phil. degree in information engineering from the Chinese University of Hong Kong in 2006, and the Ph.D. degree in electrical and computer engineering from the University of Texas at Austin in 2009.

He is an Assistant Professor with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. He has coauthored the books *Fundamentals of LTE* (Prentice-Hall, 2010), and *Stochastic Geometry Analysis of Multi-Antenna Wireless Networks* (Springer, 2019). His research interests include wireless communications and networking, mobile edge computing and edge learning, distributed learning and optimization, and big data analytics. He is a co-recipient of the 2019 IEEE Communications Society and Information Theory Society Joint Paper Award, the 2016 Marconi Prize Paper Award in Wireless Communications, and the 2014 Best Paper Award for the *EURASIP Journal on Advances in Signal Processing*. He also received the 2016 IEEE ComSoc Asia–Pacific Best Young Researcher Award. He is an Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON COMMUNICATIONS, and the *Journal of Communications and Information Networks*.

Khaled B. Letaief (Fellow, IEEE) received the B.S. degree (with Distinction), M.S., and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA.

He is the New Bright Professor of Engineering and the Chair Professor with the Hong Kong University of Science and Technology (HKUST). He is also a Distinguished Scientist with Peng Cheng Laboratory, Shenzhen. He served as consultants for different organizations including Huawei, ASTRI, ZTE, Nortel, PricewaterhouseCoopers, and Motorola. Since 1993, he has been with HKUST, where he has held many administrative positions, including the Head of the Electronic and Computer Engineering Department and the Founding Director of the Huawei–HKUST Innovation Laboratory. He also served as the Dean of engineering. Under his leadership, HKUST School of Engineering dazzled in international rankings (rising from #26 in 2009 to #14 in the world in 2015 according to QS World University Rankings). He is an internationally recognized leader in wireless communications and networks with research interest in machine learning, mobile edge computing, 5G systems and beyond. In these areas, he has over 620 journal and conference papers with over 34 300 citations. He also has a large number of patents, including 11 U.S. inventions and has made six technical contributions to international standards.

Dr. Letaief is the recipient of many distinguished awards, including the 2019 Distinguished Research Excellence Award by HKUST School of Engineering (Highest research award); the 2019 IEEE Communications Society and Information Theory Society Joint Paper Award; the 2018 IEEE Signal Processing Society Young Author Best Paper Award; the 2017 IEEE Cognitive Networks Technical Committee Publication Award; the 2016 IEEE Marconi Prize Award in Wireless Communications; the 2011 IEEE Harold Sobol Award; and the 2010 Purdue University Outstanding Electrical and Computer Engineer Award. He is well recognized for his dedicated service to professional societies and in particular IEEE where he has served in many leadership positions. These include IEEE Communications Society Vice-President for Conferences and Vice-President for Technical Activities. He also served as President of the IEEE Communications Society from 2018 to 2019, the world's leading organization for communications professionals with headquarter in New York City and members in 162 countries. He is the founding Editor-in-Chief of the prestigious IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and has been involved in organizing many flagship international conferences. He is also recognized by Thomson Reuters as an ISI Highly Cited Researcher. He is a Fellow of HKIE.