Distributed Artificial Intelligence Empowered by End-Edge-Cloud Computing: A Survey

Sijing Duan, *Student Member, IEEE*, Dan Wang, *Student Member, IEEE*, Ju Ren, *Senior Member, IEEE*, Feng Lyu, *Senior Member, IEEE*, Ye Zhang, *Member, IEEE*, Huaqing Wu, *Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—As the computing paradigm shifts from cloud computing to end-edge-cloud computing, it also supports artificial intelligence evolving from a centralized manner to a distributed one. In this paper, we provide a comprehensive survey on the distributed artificial intelligence (DAI) empowered by end-edge-cloud computing (EECC), where the heterogeneous capabilities of on-device computing, edge computing, and cloud computing are orchestrated to satisfy the diverse requirements raised by resource-intensive and distributed AI computation. Particularly, we first introduce several mainstream computing paradigms and the benefits of the EECC paradigm in supporting distributed AI, as well as the fundamental technologies for distributed AI. We then derive a holistic taxonomy for the state-of-the-art optimization technologies that are empowered by EECC to boost distributed training and inference, respectively. After that, we point out security and privacy threats in DAI-EECC architecture and review the benefits and shortcomings of each enabling defense technology in accordance with the threats. Finally, we present some promising applications enabled by DAI-EECC and highlight several research challenges and open issues toward immersive performance acquisition.**

***Index Terms*—Distributed artificial intelligence, end-edge-cloud computing, network computing, federated learning.**

Manuscript received 27 March 2022; revised 10 September 2022; accepted 20 October 2022. Date of publication 1 November 2022; date of current version 24 February 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF0604502; in part by the National Natural Science Foundation of China under Grant 62122095, Grant 62002389, Grant 62072472, and Grant U19A2067; in part by the Grant from the Guoqiang Institute, Tsinghua University; in part by the Natural Science Foundation of Hunan Province, China, under Grant 2020JJ2050 and Grant 2021JJ20079; in part by the Young Elite Scientist Sponsorship Program by CAST under Grant YESS20200238; in part by the Young Talents Plan of Hunan Province of China under Grant 2021RC3004; in part by the 111 Project under Grant B18059; and in part by the Central South University Innovation-Driven Research Programme under Grant 2023CXQD029. *(Corresponding authors: Ju Ren; Feng Lyu.)*

Sijing Duan, Dan Wang, and Feng Lyu are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: duansijing@csu.edu.cn; wang_dan113@csu.edu.cn; fenglyu@csu.edu.cn).

Ju Ren is with the Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing 100084, China, and also with Zhongguancun Laboratory, Beijing 100094, China (e-mail: renju@tsinghua.edu.cn).

Ye Zhang is with the Computer School, Beijing Information Science and Technology University, Beijing 100096, China (e-mail: yezhang@bistu.edu.cn).

Huaqing Wu is with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: huaqing.wu1@ucalgary.ca).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/COMST.2022.3218527

I. Introduction

As WE step into the Internet of Things (IoT) era, the number of mobile devices is growing exponentially, leading to massive amounts of generated data. According to recent analytics reports, there will be 30.9 billion connected IoT devices in 2025 [1], and the size of data is expected to reach nearly 79.4 Zettabytes (ZB) [2]. Meanwhile, cloud computing has developed as a dominating computing paradigm that can provide on-demand and powerful computing capabilities for different applications. The increase in data and computing capabilities consequently boosts the rapid development of artificial intelligence (AI) during the past decade [3]. Besides, powered by advanced algorithms, especially deep learning (DL) algorithms, AI has achieved unprecedented success in many aspects, e.g., image processing, natural language processing, data analytics, and other intelligent services [4], with the support of cloud computing.

However, with the ever-increasing amounts of data, cloud AI is facing significant challenges to meet the delay and privacy requirements in real-time applications, such as autonomous driving, real-time video analysis, remote healthcare, etc [5]. Sending all the data to the cloud for training or inference not only incurs uncontrollable delay for data transmitting and processing, but also intensifies the risk of privacy leakage [6]. These concerns call for distributed artificial intelligence (DAI), where model training and inference can be performed in a distributed manner nearby the data sources. The application requirements also drive the computing paradigm evolving in the same way towards a new computing paradigm, named end-edge-cloud computing (EECC), as shown in Fig. 1. EECC is built on a hierarchical computing architecture consisting of massive distributed end devices and edge servers, as well as central cloud servers. By orchestrating the distributed and heterogeneous computing resources, it offers new enabling technologies to support distributed AI to provide distributed, low-latency, and reliable intelligent services. To be specific, the main benefits of EECC are five folds [7], [8]: 1) *powerful resource supports*; 2) *fast service responses*; 3) *on-demand distributed AI computing*; 4) *secure data intelligence*; and 5) *rich applications*. Besides, AI can also assist EECC to achieve

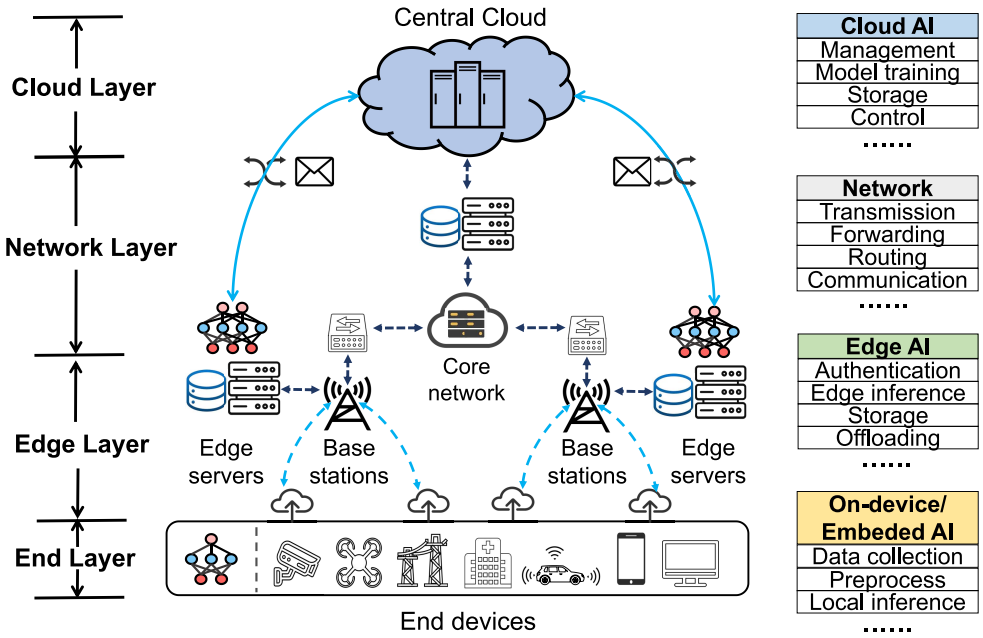


Fig. 1. The DAI-EECC architecture.

more optimal resource allocation by learning interactions with the wireless environment.

Although EECC has many benefits, there are some challenges when we use EECC to achieve distributed AI. First, frequent model parameters exchange across nodes in distributed training may lead to high latency, tremendous network resource costs, and privacy issues. How to achieve efficient distributed training empowered by EECC is a challenge. Second, it is difficult for end devices to complete highly complex model inference tasks due to constrained resources. Therefore, how to conduct distributed collaborative inference that can satisfy both model performance and communication requirements is also a challenge. Third, the EECC architecture is an open system in which heterogeneous end devices are connected to edge servers and cloud data centers. It brings a variety of security and privacy threats in terms of data storage, computing, transmission, and system management. How to design corresponding defense technologies remains to be explored. Finally, the EECC paradigm continues to have a profound impact on a wide range of intelligent applications, improving the quality of daily life and economic growth. How to deploy DAI-EECC-assisted systems in rich applications is challenging.

In this paper, we provide a comprehensive survey on the state-of-the-art distributed AI technologies empowered by EECC. Since distributed AI is based on the advance in AI technologies, we first introduce several computing paradigms and present an overview of artificial intelligence and deep neural networks. Then, we derive a holistic taxonomy for the optimization technologies of distributed AI that focus on leveraging EECC to address the challenges in distributed model training and inference, respectively. Moreover, we take the first attempt to summarize the security and privacy threats, as well as the corresponding defense technologies in DAI-EECC. Finally, we review some promising applications enabled by DAI-EECC and envision some future research directions in this promising and fast-developing field.

A. Comparison and Our Contribution

There have been some surveys investigating AI-powered cloud-edge orchestration [5], [6], [8], [9], [10], edge intelligence [12], [13], [14], [15], end-edge-cloud orchestration [7], [11], AI/ML for wireless edge network [16], [17], and privacy and security for distributed learning [18]. For example, Chang et al. [5] provided a survey of recent advances in edge computing powered by Artificial Intelligence & the Internet of Things (AIoT). They emphasized several representative AIoT applications and enabling technologies in an edge-cloud cooperation mode. Wu [6] investigated the state-of-the-art works on AI-powered cloud-edge orchestration for the IoT. Wang et al. [9] reviewed the techniques for the integration of DL and edge computing from networking, communication, and computations perspectives. Yao et al. [8] focused on cloud-edge polarization and collaboration types, including privacy-primary collaboration and efficiency-primary collaboration for personalization. Deng et al. [12] introduced edge intelligence from two aspects, i.e., AI for edge and AI on edge. The references [13], [14] provided overviews of the architectures and several key technologies for DL model computation at the network edge. Ren et al. [7] mainly discussed EECC orchestration from the perspectives of computation offloading, caching, security, and privacy. We compare this paper with the existing works in Table I, and the main differences are summarized as follows. (1) Most existing surveys focus on either the EECC orchestration or edge-cloud intelligence, while the investigation on EECC-powered distributed AI is still at its infancy stage. (2) The fundamentals summary about distributed AI computing has not been well established. (3) The privacy and security threats for distributed AI-empowered EECC are not

TABLE I
COMPARISON WITH RELATED WORKS

Ref.	Topic	EECC Orchestration	Distributed AI Fundamentals	AI-powered Distributed Learning	EECC Threats	Privacy Defense Technologies	Security Defense Technologies	EECC Empowered Applications
[5]	Edge computing for AIoT	✓	✗	✓	✗	✗	✓	✗
[6]	Cloud-Edge orchestration for AIoT	✗	✗	✓	✗	✓	✓	✓
[7]	End-Edge-Cloud orchestrated network computing paradigms	✓	✗	✗	✗	✗	✗	✗
[8]	Edge-Cloud collaborative learning	✗	✗	✓	✗	✓	✗	✓
[9]	Edge computing and DL	✓	✗	✓	✗	✓	✗	✗
[10]	AI in cloud and edge computing	✗	✗	✓	✗	✗	✗	✗
[11]	Edge-Cloud computing assisted cyber-physical systems	✓	✗	✗	✗	✓	✓	✓
[12]	DL for edge computing	✗	✗	✓	✗	✓	✗	✗
[13]	Edge intelligence and DL	✗	✗	✓	✗	✗	✗	✗
[14]	DL for edge computing	✗	✗	✓	✗	✗	✗	✗
[15]	DL for edge computing	✗	✗	✓	✗	✗	✗	✗
[16]	AI for wireless network	✗	✗	✓	✗	✓	✓	✗
[17]	ML at the network edge	✗	✗	✓	✗	✗	✗	✓
[18]	Privacy and security for distributed learning	✗	✗	✓	✗	✓	✓	✗
This paper	Distributed AI and EECC	✓	✓	✓	✓	✓	✓	✓

comprehensively summarized. (4) Most existing works mainly focus on applications with two-tiers architecture (e.g., cloud-edge, edge-end), and more research endeavors are needed to support the EECC-empowered applications. Therefore, in this paper, we aim to fill this gap by comprehensively discussing distributed AI architecture/algorithms, summarizing security and privacy threats and defenses, and introducing some representative distributed AI systems/applications. We believe that it can help researchers to quickly grasp an overview of the recent advances and find some valuable research insights in this field of study. The key contributions of this paper are highlighted as follows:

- We present an overview of preliminaries on DAI-EECC in terms of AI and DNNs, distributed learning algorithms, frameworks, modes, mainstream computing paradigms, and the benefits of the EECC paradigm in supporting distributed AI.
- We derive a holistic taxonomy for the state-of-the-art optimization technologies to conduct distributed training and inference, respectively, and present to the readers a comprehensive description and discussion of existing solutions explored in the distributed AI literature, as well as several lessons learned.
- We justify security and privacy threats in DAI-EECC architecture in categories and summarize the benefits and shortcomings of each enabling defense technology in accordance with the threat. We also point out some lessons learned.
- We review some promising applications in DAI-EECC with a variety of domains, such as the smart industry and manufacturing, intelligent transportation, smart grid, smart healthcare, and real-time video analytics. Two lessons learned are presented.
- We highlight several research challenges and open issues toward the better performance of the integration of the distributed AI and EECC paradigm to pave the way for future research on DAI-EECC and its applications.

B. Structure of the Survey

The remainder of this paper is organized as follows (illustrated in Fig. 2). We elaborate on fundamental technologies related to distributed AI, discuss the evolution of the computing paradigm, and present the EECC architecture and its benefits in Section II. In Section III, we derive a taxonomy for the state-of-the-art optimization technologies in distributed training and inference empowered by EECC, respectively. In Section IV, the security and privacy threats are identified and the corresponding defense technologies are summarized. After that, several promising applications based on DAI-EECC are introduced in Section V. Finally, we highlight some potential research directions in Section VI and conclude this paper in Section VII. The common abbreviations are presented in Table II.

II. COMPUTING PARADIGMS AND FUNDAMENTALS OF DISTRIBUTED AI

A. Computing Paradigms

We firstly introduce several main computing paradigms emerging in recent years.

1) Cloud Computing: Cloud computing (CC) [19] is a representative centralized computing paradigm, which was proposed in 2006 by Google. By deploying large data centers that comprise thousands of server units, users can access sufficient computation resources online and only need to pay for the cloud services on demand. Due to the powerful computing

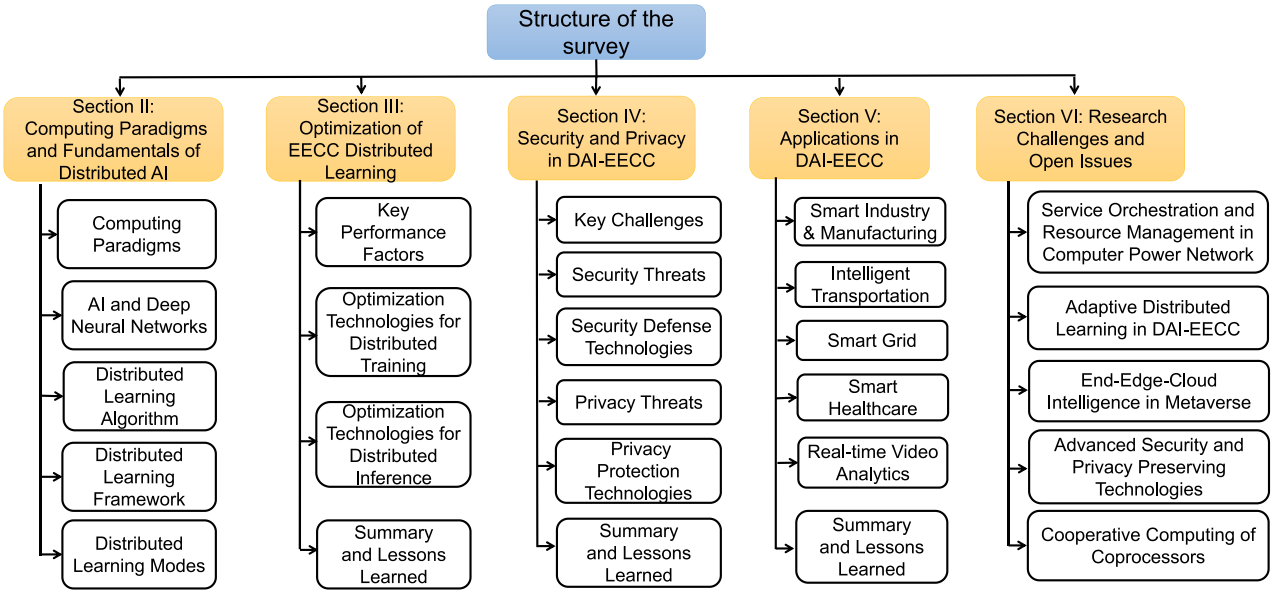


Fig. 2. The overview of paper organization.

TABLE II
LIST OF COMMON ABBREVIATIONS

Abbreviation	Description
IoT	Internet-of-Things
AI	Artificial Intelligence
DAI	Distributed Artificial Intelligence
EECC	End-Edge-Cloud Computing
AIoT	Artificial Intelligence and Internet of Things
ML	Machine Learning
DL	Deep Learning
PS	Parameter Server
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
CC	Cloud Computing
EC	Edge Computing
EI	Edge Intelligence
SGD	Stochastic Gradient Descent
Sync-SGD	Synchronous SGD
Asyn-SGD	Asynchronous SGD
GoSGD	Gossip SGD
SC	Splitting Computing
GC	Gradient Compression
HA	Hardware Acceleration
SA	Software Acceleration
GS	Gradient Sparsification
GQ	Gradient Quantization
FL	Federated Learning
CFL	Centralized Federated Learning
DFL	Decentralized Federated Learning
WP	Weights Pruning
MQ	Model Quantization
DP	Differential Privacy
LDP	Local Differential Privacy
MPC	Secure Multi-Party Computation
HE	Homomorphic Encryption
ADAS	Autonomous Driving Assistance Systems

capabilities of cloud servers, model training is usually performed in the cloud. In the past few years, CC lays the foundation for the development and popularization of AI applications, and cloud intelligence has attracted much attention from many cloud service providers, e.g., Amazon, Google, IBM, Microsoft, etc. CC empowers many famous AI projects such as Amazon Recognition, Google Cloud Vision, IBM Watson, and Microsoft Cognitive Services. However, cloud-based services are typically time-consuming and resource-demanding due to the transmission of massive data from the end to the remote cloud.

2) Edge Computing: With the development of CC, sending data to the cloud was a prominent trend in the past decades. However, the dramatically-increasing devices and data traffic in the IoTs era are posing severe burdens on the capacity-limited Internet, leading to uncontrollable service latency. It becomes difficult to meet the delay-sensitive and context-aware service requirements with CC alone. Facing these challenges, the computing paradigm is shifting from the centralized CC to distributed edge computing (EC). EC leverages the distributed resources at the network edge (e.g., router, network gateway, and base station) to deliver timely and context-aware services, providing a complement for CC to make it only responsible for delay-insensitive, resource-intensive, or computationally-complex tasks. EC improves user experiences, reduces network latency, and alleviates the load over the core network.

More recently, the concept of EC extends to cloudlet, fog computing [20] and mobile edge computing [21]. And there is an urgent need to fuse AI and EC to fully unleash the potential of edge big data. To meet this demand, edge intelligence (EI) has been widely recognized as a promising solution. EI aims to fully exploit the available resources across mobile terminals and edge nodes to perform model training and inference, providing timely and resource-efficient services for AI applications. Yet, compared to CC, edge nodes still have limitations on achieving real-time intelligent services due to limited resources and functional scalability. Therefore, the EC and CC are simultaneously required for a better quality of services in various AI application scenarios.

3) End-Edge-Cloud Computing: Motivated by the advances of emerging EC, we are expecting a hierarchical computing architecture, i.e., EECC, which can revolutionize the existing computing paradigms. As shown in Fig. 1, EECC integrates central cloud servers, numerous edge servers, and a huge number of distributed end devices, fully exploiting the differentiated capabilities of heterogeneous devices/facilities to meet the different service requirements [22]. With the support of EECC architecture, distributed AI has developed rapidly and is widely applied in various fields. In DAI-EECC architecture, from bottom to top are the end layer, edge layer, and cloud layer, where each layer is responsible for different AI tasks. The end layer can collect, pre-process or analyze data and make early decisions. The pre-processed data from the end layer can be aggregated in the edge layer for deep analysis and decision, and the cloud layer can coordinate the end layer and the edge layer.

- End Layer includes millions of widely deployed sensors, mobile devices, smartphones, and actuators, which typically have certain computing and storage capacities. End devices can be equipped with not only basic functions (such as perception and data collection) but also intelligent operations, e.g., data pre-processing and local deep neural network (DNN) inference. With local processing, only compact processed data needs to be transmitted to the upper computing devices (edge or cloud nodes). This can significantly reduce latency and network costs, which is critical for data-intensive distributed AI applications. Due to the computing and storage resource limitations, only shallow DNN models can be executed on end devices. Thus, model compression and acceleration technologies for resource-constrained devices have attracted much attention, which will be reviewed in Section IV-C. For computation-intensive tasks, the end devices need to resort to edge and cloud resources for further processing.

- Edge Layer is the bridge that connects the end layer and cloud layer, including base stations, routers, IoT gateways, access points, etc [23]. The edge layer is mainly responsible for functions such as authentication, edge inference, model selection, and request forwarding. Specifically, it will authenticate and authorize the trusted device for security cooperation. The edge nodes can receive and store the data from end devices, conduct edge training or inference, and deliver the final results to the end devices or offload the intermediate results to the cloud servers. Since the edge nodes are closer to data sources than the cloud data centers, edge processing can reduce the service latency and provide auxiliary computing resources for computation-intensive intelligent applications.

- Cloud Layer is equipped with powerful computing capabilities and abundant storage resources, cloud data centers can provide on-demand services to empower various intelligent applications. Generally, the cloud layer is responsible for data storage, device management, controlling, complicated model training, etc. Particularly, the cloud layer can train AI models with high accuracy due to the sufficient data and resources. Moreover, the cloud layer can coordinate the end layer and edge layer. It receives data or intermediate results from end layer and edge layers, empowers efficient distributed AI computing, and provides powerful resource supports.

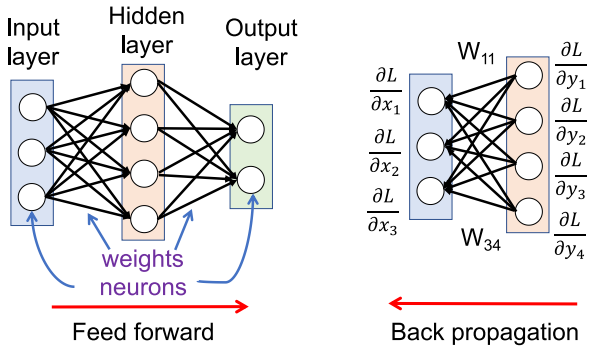


Fig. 3. The example of DNN computation.

However, traditional distributed AI algorithms have difficulties satisfying the high requirements for real-time data processing due to high communication latency. Meanwhile, the direct integration of distributed AI and EECC is not straightforward since heterogeneous resource orchestration is not considered in traditional distributed AI. Therefore, it motivates us to design new collaborative training and inference schemes that can adaptively incorporate heterogeneous resources of diverse compute nodes, so as to offer flexible communication and computing services in an efficient manner.

B. Artificial Intelligence and Deep Neural Networks

Artificial Intelligence (AI) is a general term that implies the use of a computer to model intelligent behaviors with minimal human intervention. Deep neural networks (DNNs), also referred to deep learning (DL), are a representative part of the broad field of AI, which has the ability to automatically discover and learn effective features from raw data. Nowadays, DL has been extensively applied in many applications such as computer vision, natural language processing, and speech recognition [24]. A conventional DL model is composed of three layers: input layer, hidden layer, and output layer. Fig. 3 shows a simple DL network example. The neurons in the input layer receive some values and feed-forward them to the neurons in the hidden layer, the weighted sums from one or more hidden layers are ultimately propagated to the output layer, which presents the final outputs of the network to the user [25]. At each layer, the general computation is $y_j = f(\sum_i^n w_{ij} \times x_i + b)$, where w_{ij}, x_i and y_j are the weights, input activations and output activations, respectively, $f(\cdot)$ is a non-linear function, and b is the bias term. Some activation functions include the ReLu and Softmax functions. Finally, the errors that are derived by loss function and label backpropagate through the network for weight update $w_{ij}^{t+1} = w_{ij}^{t} - \alpha \frac{\partial L}{\partial w_{ij}}$ in the next iteration $t+1$, where α is the learning rate.

DNN model learning involves two stages, i.e., training and inference. The objective of the training is to optimize the weights of the network such that the loss function is minimized. During training, the weights are calibrated through stochastic gradient descent (SGD). After training, running the program with these weights is referred to as inference. With the rapid development of DL, a large number of new models and architectures have emerged, e.g., convolutional

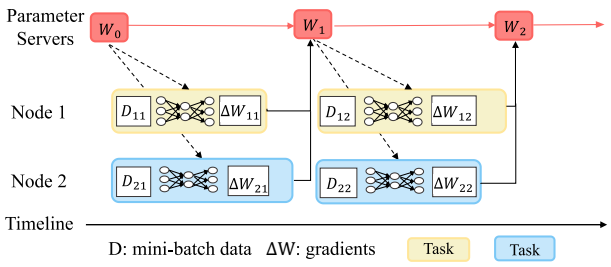


Fig. 4. Synchronous SGD.

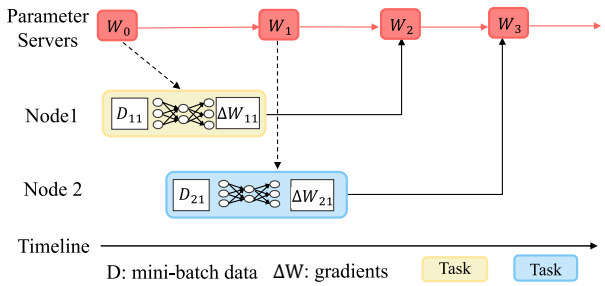


Fig. 5. Asynchronous SGD.

neural network (CNN), recurrent neural network (RNN), auto-encoder, capsule network, transfer learning, and generative adversarial network (GAN).

C. Distributed Learning Algorithm

For most DL applications, given a dataset $\mathcal{X} = \{x_i : i = 1, \ldots, |\mathcal{X}|\}$, the goal is to learn the parameters θ of a model with respect to an empirical loss function f, defined as $f(\theta) \triangleq \frac{1}{\mathcal{X}} \sum_{i=1}^{|\mathcal{X}|} F(x_i; \theta)$, where $F(x_i; \theta)$ is the loss with respect to a datapoint x_i and the model θ. However, considering the limited resources such as computing power, memory, and storage of a single device, the data samples or DNN model is first split into several parts before training. Particularly, in distributed AI training process, N worker machines are used to be in charge of computing stochastic gradients, and then are sent to the parameter server j for distributed stochastic optimization. Each parameter server j is responsible for storing a subset $\theta[j]$ of the model, and performing updates on $\theta[j]$ [26]. All worker machines compute in a parallel manner. In general, there are three main parallelism technologies, i.e., data parallelism, model parallelism, and hybrid parallelism, which will be introduced in detail in Section II-E. The training algorithms are introduced in the following, including synchronous SGD, asynchronous SGD, and gossip SGD.

- *Synchronous SGD (Sync-SGD):* Fig. 4 shows the Sync-SGD setting, where all computing nodes wait for other nodes to complete the gradient computation in each round during training. Then the PS accumulates gradients and generates a new global parameter update. Finally, the computing nodes download the updated model and begin the next iteration.

Although the Sync-SGD can guarantee convergence, due to the tightly coupled nature of the nodes, the computing speed will be slowed down by the stragglers, resulting in a long convergence time [27]. To address this problem, one possible solution is to aggregate partial gradients. Chen et al. [26] proposed a distributed training acceleration scheme called synchronous optimization with backup workers. It only aggregates a subset of gradients during the process of gradient aggregation to prevent the stragglers from slowing down the overall computational efficiency, while using backup workers to avoid any staleness in gradients. However, this method is not efficient if existing persistent stragglers, as a portion of data will be lost. To enhance robustness to persistent stragglers, instead of discarding the stragglers, Ferdinand et al. [28] fixed the duration of the computational epoch, and weighted each update by the amount of work that is finished at the end of each epoch. The global model is the weighted aggregation of each update. This work is robust to both persistent and non-persistent stragglers, and compared to [26], it has a faster convergence speed.

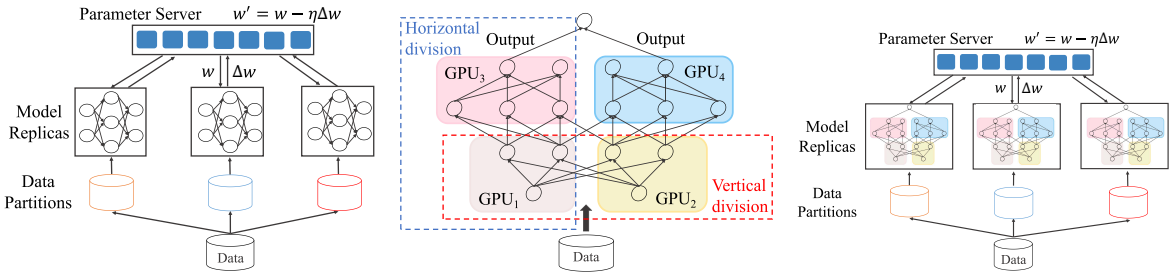
Algorithm 1: The Pseudocode of Asyn-SGD

Input: Data set $\mathcal{X}$, mini-batch size B, learning rates $\gamma_0, \gamma_1, \ldots$; decay rate α; number of mini-batches to aggregate N; model initialization $\theta^{(0)}$.

```
// [worker k]
while True do
    Read θ̂_k = (θ[0], ..., θ[M]) from PS
    G_k^(t) := 0
    for i = 1, ..., B do
        Sample data point x̃_i from X
        G_k^(t) ← G_k^(t) + (1/B) ∇F(x̃_i; θ̂_k)
    end
    Send (G_k^(t)) to parameter servers
end
// [parameter server j]
for t = 0, 1, ... do
    Wait for gradient G from any worker
    θ^(t+1)[j] ← θ^(t)[j] − γ_t G[j]
    θ̄^(t)[j] = α θ̄^(t−1)[j] + (1 − α) θ^(t)[j]
end
```

- *Asynchronous SGD (Asyn-SGD):* Asyn-SGD can improve the iterative speed of distributed AI training by removing the coupling relationship and reducing the dependence between nodes. In the Asyn-SGD setting, as shown in Fig. 5, each node independently computes gradients and updates the model. The pseudocode for Asyn-SGD is given in Algorithm 1.

Compared to Sync-SGD, the asynchronous version is more robust to the single point of failure, i.e., if the PS fails, other nodes can continue to process and avoid the problem of the synchronization barrier. Since there is no synchronization between nodes when updating global model parameters, some nodes might compute gradients using outdated global model weights, making model convergence slow and unguaranteed. Therefore, Asyn-SGD may have problems with stability and accuracy due to the staled or delayed weight updates. To address this problem, Zhang et al. [29] proposed a variant of the Asyn-SGD algorithm where the learning rate is adjusted according to the gradient staleness. Besides, Zheng et al. [30] designed delay compensated Asyn-SGD, which leveraged



(a) Data Parallelism (b) Model Parallelism (c) Hybrid Parallelism

Fig. 6. Several types of parallelism technologies.

Taylor expansion of the gradient function and efficient approximation to the Hessian matrix of the loss function. Although Asyn-SGD can accelerate the training speed by eliminating costly synchronization, the current convergence of coordinate descent algorithms is based on somewhat unrealistic assumptions. In particular, the age of the shared optimization variables being used to update a block is assumed to be independent of the block being updated. Sun et al. [31] proved the convergence of asynchronous-parallel SGD under more realistic assumptions, e.g., permitting both the deterministic cyclic and random rules for block choices.

- *Gossip SGD (GoSGD):* Although Sync-SGD and Asyn-SGD can achieve gradient exchange between distributed nodes. The huge communication overhead hinders the scalability of large-scale distributed training [32]. Inspired by [33], a more communication-efficient distributed training approach based on a randomized gossip algorithm can accelerate distributed training, which is called Gossip SGD (GoSGD) [34]. GoSGD trains DNN models by exchanging parameter information among multiple threads in a peer-to-peer way. In particular, the GoSGD algorithm considers M independent workers, each of them hosts a CNN of the same architecture with a set of weights noted x_i for worker i. The GoSGD includes two steps: gradient update and mixing update. At each iteration, every worker node performs SGD on its data in the gradient update step, and then sends its weights to other randomly chosen nodes in the mixing update step. In GoSGD, each node can receive information from multiple nodes but can only send its updates once. GoSGD can retain the advantages of both Sync-SGD and Asyn-SGD by replacing the all-reduce collective operation of Sync-SGD with the gossip aggregation method [35].

To further improve the communication efficiency of GoSGD, the authors realized [36] a linearly convergent gossip algorithm with compressed communication. Chen et al. [32] added periodic global averaging into GoSGD, to mitigate the slow convergence rate in distributed training. Besides, Assran et al. [37] combined GoSGD with stochastic gradient updates. This method can accelerate the training of DNNs by reducing communication overhead and mitigating the effects of stragglers.

D. Distributed Learning Framework

Recently, some emerging distributed learning frameworks are developed, making large-scale distributed AI easy and flexible for developers and researchers. These frameworks can be deployed in high-performance GPU-based servers and extensively applied in various AI applications. Table III summarizes mainstream distributed AI training frameworks, including Tensorflow, Pytorch, Keras, etc. These frameworks integrate different parallelism mechanisms.

E. Distributed Learning Stages

Distributed learning includes two stages, i.e., training and inference.

1) Training Stage: Distributed AI performs distributed training by utilizing parallelism technologies, where learning agents may be located in different geographical locations. Parallelism technologies enable the system to make full use of all distributed computing resources. Each separated computing node executes parts of intelligent tasks, which is helpful to achieve rapid analysis and computation. Compared to centralized AI, distributed AI has many intrinsic advantages: 1) multi-nodes collaboration for achieving rapid training process of complex models; 2) strong scalability for complex scenarios; and 3) avoiding a single point of failure. Thus, distributed AI is suitable for IoT and mobile Internet applications. In the following, we introduce several main parallelism technologies.

- *Data Parallelism:* For most DL applications, the data set is too big to be computed on a single device. Data parallelism is a feasible solution that partitions data and distributes each batch to multiple distributed computing nodes, as shown in Fig. 6(a). To achieve data parallelism, model averaging, Sync-SGD, and Asyn-SGD algorithms have been widely used in training large-scale DNNs. With data parallelism, data samples can be assigned across the three-tiers architecture of the end devices, edge servers and cloud server [53], thus the computation speed of real-time applications can be accelerated.

- *Model Parallelism:* Compared to data parallelism, model parallelism is more common in DAI-EECC. It is an approach that splits the model into several sub-models and places them on multiple nodes (or GPUs), where each node is responsible for parts of the model training [54]. As depicted in Fig. 6(b), different layers and neurons of the DNN model are deployed in multiple GPUs. For model parallelism, each GPU cannot update independently and needs to cooperate with other nodes. According to the structural characteristics of neural networks, model parallelism methods can be categorized into horizontal

TABLE III
AN OVERVIEW OF POPULAR DISTRIBUTED LEARNING FRAMEWORKS

Framework	Owner	Support language	Mobile available	Highlight
Tensorflow [38]	Google	Python/C++/Haskell/Java GO/Rust/C#/R/Scala	✓	1. End-to-end open source ML framework. 2. Providing distributed training APIs to support training across multiple machines.
Pytorch [39]	Facebook	Python/C++/Julia	✓	1. Open-source ML framework. 2. Supporting training across multiple machines.
PaddlePaddle [40]	Baidu	Python/C++/JAVA	✓	1. End-to-End open source DL platform. 2. Providing API to perform distributed training.
MindSpore [41]	Huawei	Python/C++/Julia/Java	✓	1. Open source DL platform. 2. Supporting mobile, edge and cloud scenarios.
MegEngine [42]	MEGVII	Python/C++	✓	1. Open source DL framework. 2. Inference efficiently on all platforms.
Horovod [43]	Uber	Python/C++	✓	1. Fast distributed DL framework in Tensorflow. 2. Employing efficient inter-GPU communication via ring reduction.
CNTK [44]	Microsoft	Python/C++	✓	1. Unified DL toolkit that describes DNN as a series of computational steps via a directed graph.
MACE [45]	Xiaomi	Python/C++	✓	1. DL inference framework optimized for mobile heterogeneous computing platforms.
DL4J [46]	IBM	Java/Scala/Python/ Clojure/Kotlin	✓	1. Open-source DL library for JVM. 2. Integrating with Hadoop and Spark.
JAX [47]	Google	Python/C++/Java	✓	1. A simplified library for TensorFlow. 1. Combining Autograd and accelerated linear algebra.
BytePS [48]	ByteDance	python	✗	1. High performance and generic framework for distributed DNN training. 2. Supporting various DL frameworks.
Caffe2 [49]	Facebook	Python/C++	✓	1. Lightweight and modular ML framework. 2. Supporting major hardware and power one of the largest deployments of mobile DL.
Keras [50]	F.Chollet	Python	✗	1. Open-source ML library. 2. Containing many implementations of commonly used DNN building blocks.
Flux [51]	Mike Innes	Julia	✓	1. Open-source ML software library. 2. Supporting a layer-stacking-based interface for simpler models.
PlaidML [52]	Intel	Python/C++	✗	1. Portable tensor compiler. 2. Enabling DL on devices where the available computing hardware is either not well supported.

division by layer, vertical division by layer [55], and random model division [56].

- *Horizontal Division by Layer* refers to the situation where a working node is responsible for the computation of one or multiple layers, and a computation mode pipeline is formed between nodes to propagate errors and gradient descent among neural network layers, e.g., GPU_1 and GPU_3 in Fig. 6(b) are horizontal division. This type of division is suitable for neural networks with many layers.
- *Vertical Division by Layer* is applicable to the situation where there are many neurons in a single neural network layer. Particularly, neurons at the same layer are assigned to distributed working nodes, and a single node needs to wait for the neighboring nodes to complete the computation and propagate gradient updates to each other, e.g., GPU_2 and GPU_4 in Fig. 6(b) are vertical division.
- *Random Model Division.* For large neural networks, the communication cost of horizontal and vertical division is very high, which causes a severe bottleneck for the training acceleration. An attempt to address this problem is random model division. The core idea is to select a small-scale sub-network (called skeleton network) from the original large-scale network, and then update the nodes in the skeleton network according to the model performance during the training process until convergence, e.g., GPU_1, GPU_2, GPU_3, and GPU_4 in Fig. 6(b) are random model division.

In the EECC architecture, a large DNN model can be divided into several parts and executed across the mobile devices, edge servers and cloud servers [57]. When training the DL models in a distributed manner, the frequent gradient exchanges across the nodes inevitably consume tremendous network resources. Thus, gradient compression can be used to reduce communication costs, i.e., gradient sparsification [58], [59] and gradient quantization [60], [61], [62].

- *Hybrid Parallelism:* Data parallelism and model parallelism are effective methods in distributed computing. However, as the increasing sizes of the DNN model and the available data set, the training process becomes more complex and computationally intensive, which usually takes a longer time to complete [63]. To this end, several research works explored the hybrid parallelism approach by combining both

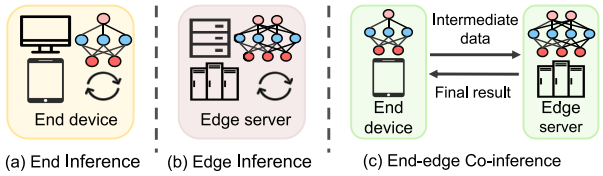


Fig. 7. The main model inference modes.

data and model parallelism for efficient distributed training. As depicted in Fig. 6(c), each data-parallel worker consists of multiple devices to accelerate the training process by exploiting model parallelism [64]. For example, Geng et al. [65] incorporated vertical data parallelization and horizontal model parallelization, and proposed a novel framework to achieve higher performance and better scalability for large-scale distributed AI. Nowadays, hybrid parallelism has been applied in a hierarchical AI learning framework, which can adaptively assign the DNN model layers and data samples across three levels of the mobile device, edge server, and cloud data center [53].

2) Inference Stage: To improve the performance of intelligent applications, an effective approach is to conduct distributed inference through orchestrating cloud, edge, and end resources. In this section, we discuss the distributed inference driven by EECC, including the model inference modes, key performance indicators, and several enabling technologies.

According to the position of DNN model inference, we summarize several main inference modes as follows, which is shown in Fig. 7.

- *End Inference:* Considering the communication latency and privacy issues, DL inference at the end layer attracts much attention. However, the end devices are typically resource-constrained while the DL models with millions of parameters require high-performance computing units, such as CPUs, GPUs, and DSP. Therefore, it is necessary to optimize the DNN models via model compression and acceleration, making the trade-off between computing resource consumption and model performance. In Section III-C, we will introduce the optimization technologies for end inference.

- *Edge Inference:* In edge inference, devices send data to the edge server, which then makes inference based on the deployed DNN model. When the inference is done, the results are returned to the devices. The edge inference can guarantee the acceptable latency for real-time applications. However, the inference performance depends on the network conditions between devices and the edge server and the computing load of the edge server.

- *End-Edge Co-Inference:* In the end-edge co-inference mode, the device firstly divides the DNN model into multiple parts and performs local model inference according to its capabilities. The remaining parts and intermediate results are offloaded to the edge server for further execution. Finally, the edge server returns the results to the device. Compared with the end or edge inference mode, this one is more flexible and can exploit distributed resources to perform computation [66].

- *End-Edge-Cloud Co-Inference:* In this mode, the DNN model is executed through end-edge-cloud synergy. The end devices are responsible for input data collection and shallow DNN inference. For a given input, the edge node will determine whether it can be successfully processed by the DL model deployed on the resource-constrained edge device. If not, it turns to the more powerful DL model deployed at the cloud [67]. Compared to end-edge co-inference, moving the inference to the cloud can access more computational power. Thereby, the integration of edge and cloud computing resources is essential [68], [69].

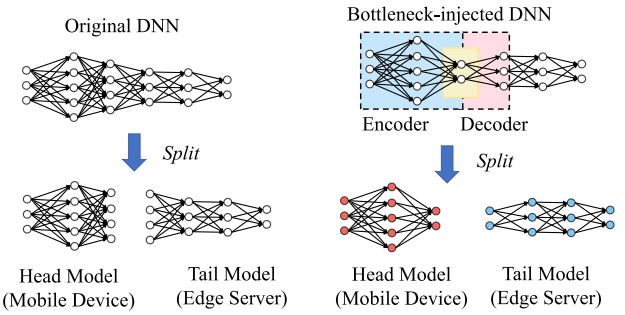
III. Optimization of End-Edge-Cloud Distributed Learning

In this section, we first introduce several key performance metrics for evaluating the optimization performance of EECC empowered distributed learning. Then, we present a holistic taxonomy for the state-of-the-art optimization technologies to conduct distributed training and inference, respectively.

A. Key Performance Metrics

We summarize several main metrics for optimization evaluation of end-edge-cloud distributed learning.

- *Accuracy:* refers to the ratio of the number of input samples that get the correct predictions from inference to the total number of input samples. DNN inference accuracy is affected by the model structure, the number of samples, and the capability of devices.
- *Latency:* refers to the time consumption for the whole inference process, including preprocessing, model inference, network transmission, and postprocessing. Some real-time intelligent mobile applications usually have stringent latency requirements. Basically, latency is affected by many factors, including network conditions, inference modes, the capability of computing devices, and so on.
- *Energy cost:* is affected by the size of the DNN model and computing resources. Particularly, the computation and communication of DNN model inference bring a large amount of energy consumption. Compared with the edge server and the cloud data center, the end devices are usually battery-limited.
- *Communication cost:* affects the inference performance significantly. It is necessary to minimize the overhead during the DNN model inference, especially the expensive wireless area network bandwidth usage for the cloud. Communication overhead mainly depends on the size of DNN models.
- *Memory footprint:* A high-precision DNN model includes millions of parameters and is hungry for hardware resources. In mobile devices, CPUs and GPUs typically compete for shared and scarce memory bandwidths. For the optimization of the distributed inference, the memory footprint is a nonnegligible indicator. Memory footprint is mainly affected by the size of the original DNN model and the way of loading the tremendous DNN parameters.
- *Privacy:* refers to the private information protection for users. With the introduction of laws and regulations, such as General Data Protection Regulation (GDPR), users are



(a) SC w/o modification (b) SC with bottleneck injection

Fig. 8. Two different splitting computing approaches.

paying more and more attention to privacy. The degree of data privacy protection in the distributed learning process also becomes an important evaluation indicator.

B. Optimization Technologies for Distributed Training

1) Splitting Computing: Splitting computing (SC) divides a large DNN into the head and tail models, respectively executed across the mobile device and edge server or cloud server. It can reduce data transfer delays by establishing a task-oriented compression scheme in DNN training [70]. Existing SC approaches include two categories: (i) *SC without network modification* and (ii) *SC with bottleneck injection*. Fig. 8 provides an overview of the approaches.

- *SC without network modification:* In this class of approaches, the architecture and weights of the head $\mathcal{M}_H(\cdot)$ and tail $\mathcal{M}_T(\cdot)$ models are exactly the same as the first ℓ layers and last $L-\ell$ layers of $\mathcal{M}(\cdot)$. This category mainly focuses on computer vision tasks such as image classification. An inevitable problem for SC without network modification is selecting an appropriate splitting point such that distributed DNN training can meet various types of cost requirements. For this problem, SC is firstly utilized in Neurosurgeon between the cloud data centers and mobile edge, aiming to search for the automatic layer partitioning strategy in the computer vision DNN model [57]. Kim et al. [71] proposed to split deep network (e.g., CNN model) into a tree of subnetworks, which can enable straightforward model parallel training across multiprocessors. Eshratifar et al. [72] considered the DNN splitting strategy under energy consumption constraints of mobile devices, the cloud server load constraints, and the service quality requirements. Optimization formulations of the forward and backward propagation were provided at layer granularity for efficient DNN collaborative computation.

More recently, Cohen et al. [73] designed a technique to code the activations of split DNN layer, which is suitable for object-detection and classification tasks across mobile phone or edge devices and cloud. However, the key problem is that naive SC approaches rely on the existence of natural bottlenecks, i.e., intermediate layers whose output tensor size is smaller than the input size. Without such natural bottlenecks in the early layers of the model, such as ResNet [74], InceptionNet [75], Faster R-CNN [76], etc., naive splitting approaches would fail to improve performance for most intelligent tasks.

- *SC with bottleneck injection:* To address the above issue, it is feasible to introduce artificial bottlenecks to DNN models by modifying their architecture. Particularly, this class of models consists of three parts: $\mathcal{M}_E$, $\mathcal{M}_D$, and $\mathcal{M}_T$. The first two sections of the modified model transform the input x into a version of the output of the ℓ-th layer via the intermediate representation z. The model is split after the first section, namely, $\mathcal{M}_E$ is the head model, and the concatenation of $\mathcal{M}_D$ and $\mathcal{M}_T$ is the tail model. The layer between $\mathcal{M}_E$ and $\mathcal{M}_T$ is the injected bottleneck. The authors in [77] and [78] firstly propose to alter existing DNN architectures by designing relatively small bottlenecks at early layers in DNN models, instead of applying compression technologies (e.g., quantization and autoencoder) to the models. Following these studies, Matsubara and Levorato [79] proposed to split the DNNs into head and tail models and train on the mobile device and edge server, respectively. Then, they introduced a bottleneck layer in the early layers of the head model to achieve in-network compression.

In summary, most of the existing SC approaches mainly focus on exploring appropriate DNN model splitting points to minimize the total delay of collaborative computing. However, there are two limitations. On the one hand, SC methods are only discussed for computer vision tasks while ignoring the combination of multiple task models. On the other hand, few studies consider the tradeoff between the accuracy of distributed training and specific indicators (such as communication delay and energy consumption), which is important in practical scenarios and deserves further study.

2) Compression: Compression is an effective optimization technology during model training [98], [99], including gradient sparsification, gradient quantization, weight pruning, model quantization, automatic compression, and lightweight model design. Particularly, gradient sparsification and quantization are designed for reducing distributed communication costs and the other technologies are used for on-device optimization. We summarize some related works in Table IV and Table V, which are categorized by gradient-based compression and model-based compression.

Gradient Sparsification (GS): GS reduces communication latency by selecting parts of gradients during aggregation, which can achieve an arbitrary compression ratio [100]. However, the large gradients and time-consuming computation limit the scalability of multi-node training. Deep gradient compression (DGC) [58] was further proposed to reduce the communication bandwidth consumption, by using Top-*k* sparsification twice. Specifically, DGC randomly selects 0.1% - 1% proportion of gradients greater than a certain threshold for Top-*k* operation. However, DGC was proposed to adopt a heuristic approach in terms of sparse threshold selection, and the two Top-*k* operations are time-consuming.

To address the above issue, the authors [59] improved the sparsity threshold selection strategy. It was assumed that the gradient follows a Gaussian distribution, and then the threshold was selected according to the certain compression ratio. Compared to DGC, the time consumption in the sparse

TABLE IV
OVERVIEW OF GRADIENT-BASED COMPRESSION

Approach	Ref.	Key idea	Compression ratio	Data set
Gradient Sparsification (GS)	[58]	Momentum correction, local gradient clipping, momentum factor masking and warm-up training.	600×	ImageNet, Cifar10, PTB
	[80]	Gradient correction and batch normalization update with local gradients.	$1-318\times$	MNIST, CIFAR-10, ImageNet
	[81]	Merging gradients from neighbor layers.	Not mentioned	ImageNet, Cifar10, PTB
	[82]	Leveraging inter-node redundancy.	386×	ImageNet, Cifar10, CamVid
	[83]	Leveraging similarity in the gradient distribution among learners.	$65-400\times$	ImageNet, WMT, SWB300
	[84]	Binarization method and optimal weight update encoding.	4×	MNIST, CIFAR-10, ImageNet, PTB, Shakespeare
Gradient Quantification (GQ)	[85]	Cosine-based quantization.	3×	MNIST, CIFAR-10, BraTS
	[86]	Dither quantization and compressive sensing.	Not mentioned	MNIST, CIFAR-10, ImageNet
	[87]	Indirect quantization via factorization.	100×	MNIST, CIFAR-10, ImageNet
	[88]	Standarding deviation ratio according to gradient mean.	$1-16\times$	Cifar-10, ImageNet
	[89]	Binary and multi-level gradient quantization.	10×	CIFAR, ImageNet
GS+GQ	[90]	Combining aggressive sparsification with quantization.	$15-20\times$	ImageNet, MNIST

TABLE V
APPROACHES TO MODEL-BASED COMPRESSION

Approaches	Ref.	Key idea	Shortcoming
Weight Pruning (WP)	[91]	Introducing intra-convolution kernel pruning and connectivity sparse from inter-convolution kernel.	System-level design is not considered.
	[92]	Combining compiler optimization, code generation, and block-based column-row pruning.	Microcontroller units are not considered.
Model Quantization (MQ)	[93]	Improving the quantization function and quantizing the batch normalization parameters by a logarithm-like method.	Performance is not evaluated on multiple hardware devices.
	[94]	Proposing an FPGA-centric mixed scheme quantization that is suitable for Gaussian-like weight distribution and fix-point computation.	Not flexible for bit-width and without automated architecture optimization.
	[95]	Proposing a sequential single path search method for mixed-precision quantization.	Hardly to access the accurate gradients for discrete quantization.
Co-Design and Automatic Compression	[96]	Proposing automated model compression by jointly applied pruning and quantization.	Only DNN models are evaluated.
	[97]	Proposing an automated framework for model compression and acceleration.	Based on heuristic method and time-consuming.

operation is significantly reduced. Yet, the strong prior condition that the gradient follows a certain Gaussian distribution may not be satisfied, which leads to differences between the actual compression ratio and the predetermined compression ratio. Recently, Abdelmoniem et al. [101] proposed sparsity-distribution-based compression, assuming that the gradient corresponds to a sparsity distribution. Then a multi-stage threshold estimation method was proposed to get the threshold accurately.

Gradient Quantization (GQ): GQ reduces the communication bandwidth by replacing the float-point gradients with low-precision values. Wen et al. [60] developed TernGrad which uses ternary gradients quantization, i.e., three numerical levels -1, 0, 1 to accelerate distributed DL. Inspired by stochastic quantization and lossless code for quantized gradients, Alistarh et al. [61] introduced quantized SGD to compress gradient updates by smoothly trading off communication bandwidth and convergence time. In [62], the authors leveraged binary gradient quantization in the proposed digital SGD at the wireless network edge to reduce the high dimension parameters, so that the gradients can be transmitted under a limited channel bandwidth.

Weights Pruning (WP): WP is a popular method for model compression by removing redundant connections while preserving accuracy. Fig. 9 shows a general framework for weight pruning. The steps are: 1) training a neural network to learn the important connections; 2) pruning the unimportant

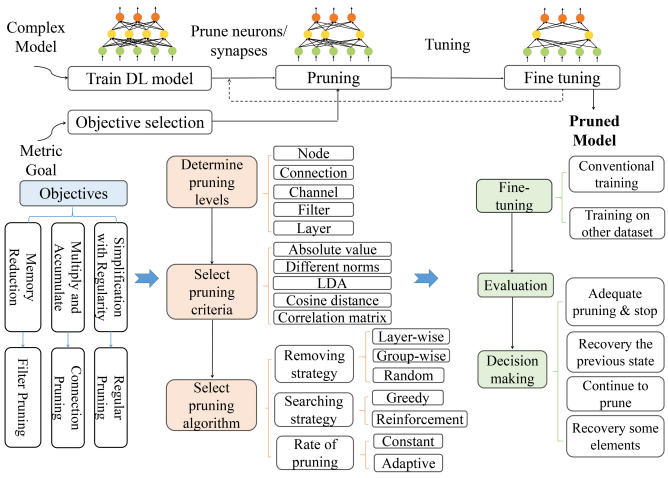


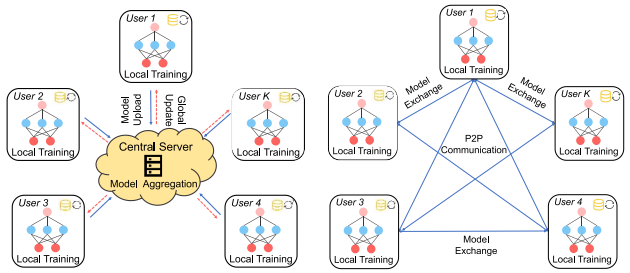
Fig. 9. General framework for weight pruning.

neuron connections; and 3) retraining the network to fine-tune the weights of the remaining connections. Particularly, we can determine the pruning level based on different objectives. The pruning criterias include absolute value, different norms, linear discriminant analysis (LDA), cosine distance, entropy [102], and correlation matrix. There are several pruning algorithms such as removing neurons/synapses, searching, etc [103]. Recently, weight pruning was used in achieving real-time DNN inference on mobile devices [91], [92], [104].

Model Quantization (MQ): MQ refers to replacing float-point numbers with low-bit numbers, thus significantly reducing memory footprint and accelerating computation. Examples of MQ include binary quantization [105] and ternary quantization [106]. However, using fixed number types may lead to a sub-optimal result. To this end, Oh et al. [107] introduced number abstract data type and proposed an automatic quantization framework for DNN. Specifically, this framework encapsulates the internal representation of a number and then reconstructs a novel representation (type, bit width, and bias) for the number. Compared to the 32-bit float-point numbers, it can reduce the parameter size up to 16x with 1% loss of accuracy. Since model quantization helps to reduce the model size and computation latency, it has been successfully applied in many applications of mobile phones [93] and embedded devices [94]. However, due to the constraints of hardware resources, delay, and energy consumption, it is difficult to quickly determine the quantization bit accuracy of each layer in mobile DL networks [95].

Co-Design and Automatic Compression: Although both pruning and quantization methods are effective, sometimes a single compression technique may not suffice to meet the diverse requirements. Thus, the combination of different compression technologies helps achieve a fast and accurate DNN model for mobile applications, such as Deep Compression [108], Octave Deep Compression [109], and Joint Pruning & Quantization [110]. However, conventional model compression technologies rely on hand-crafted heuristics and rule-based strategies, requiring domain experts to explore large design space, which is usually sub-optimal and time-consuming. To this end, some researchers studied automated model compression strategies by leveraging reinforcement learning, which can exceed the performance of rule-based model compression methods [96], [111]. To optimize compression and acceleration of DNN on mobile devices and help users achieve local and efficient data processing, Tencent designed PocketFlow [97], an automated DL model compression framework, to support model pruning & quantization co-design.

Lightweight Model Design: DNN usually improves DL network performance by broadening network structure. However, it leads to problems such as large demand for memory and CPUs/GPUs, slow inference speed, and high power consumption, making it difficult to apply to real-time scenarios and mobile devices. For example, the slow running speed of the neural network for autonomous driving may cause accidents, and the high power consumption will deteriorate the endurance of mobile terminals such as cars and mobile phones. The lightweight neural network is a feasible solution by designing DNN models with limited memory and computational complexity while guaranteeing high accuracy. There are several mainstream models including MobileNet V1 [112], V2 [113], V3 [114], SqueezeNet [115], ShuffleNet [116], Xception [117], GhostNet [118] and PP-LCNet [119]. By adopting state-of-the-art technologies, e.g., depth-wise separable convolution, point-wise group convolution, channel shuffle, and fire module, these lightweight models significantly promote efficient inference on-device [120].



(a) Centralized FL (CFL) (b) Decentralized FL (DFL)

Fig. 10. The taxonomy of FL with network architecture.

3) Federated Learning: Federated learning (FL) is a promising technique to preserve data privacy when training DL models on distributed clients [134], [135], [136], [137]. Rather than uploading the raw data to a central data center, FL leaves local training data on edge devices (or mobile devices) and learns a shared model by aggregating locally-computed updates to a global DL model [138], [139], [140], [141]. Specifically, the communication process contains four steps: (i) a fraction of selected devices download a global DL model from the central server at the beginning of each round; (ii) each client performs local computation based on the global model and its own local data set; (iii) each client uploads the updated parameters to the server; and (iiii) the central server aggregates the client updates. This process repeats until the model converges. From the network architecture perspective, FL can be divided into two categories, i.e., centralized FL and decentralized FL [142], as shown in Fig. 10. In this section, we discuss the taxonomy of classical FL and summarize related works in Table VI.

Centralized FL (CFL): CFL is the most commonly used architecture in FL systems, including a centralized server and a set of distributed clients to perform model training. In the initial CFL, the cloud server acts as an aggregation node for coordinating the aggregation and distributing the model updates to the clients. However, the communication between clients and the cloud server is high-latency and unpredictable [134]. With the emergence of EC technologies, edge servers can serve as parameter servers to provide storage and computing resources for FL clients at the network edge [143], which decreases communication costs and latency. Nevertheless, one disadvantage is the limited number of participant clients that each server can access, leading to inevitable performance degradation.

It is necessary to investigate the orchestration of end devices, edge servers, and the cloud center. Fig. 11 shows an end-edge-cloud hierarchical FL paradigm, which has two main benefits: (i) improving model performance by accessing lots of training samples in the cloud server, and (ii) achieving efficient communication with local clients via edge servers. Compared with cloud-based and edge-based FL, hierarchical FL achieves a good tradeoff between communication costs and model training performance [121], [144], [145]. However, there are still challenges in implementing a hierarchical FL system. Specifically, the convergence guarantee of distributed algorithms is unknown and needs to be further studied. In addition,

TABLE VI
THE TAXONOMY OVERVIEW OF FL

Reference	Key idea	Optimization objective
[121], [122]	Tradeoff between communication costs and model training performance.	Accuracy, communication overhead
[123]	Personalized FL with data, statistical, and model heterogeneity.	Accuracy, latency
[124]	Utilizing distributed Adam optimisation and compression of uploaded models to improve communication efficiency.	Communication overhead
[125], [126]	More frequent local aggregations at the edge servers and fewer aggregation in the cloud server.	Communication overhead, latency
[127]	Deriving a general convergence bound for hierarchical SGD with non-IID data, non-convex objective functions.	Accuracy
[128]	Intra-cluster communication and hierarchical aggregation between the clusters and cloud server.	Communication overhead
[129]	Exploiting edge aggregation to avoid frequent communications with cloud servers.	Communication overhead
[130]	Identifying irrelevant updates and precluding them from being uploaded for reduced network footprint.	Accuracy, communication overhead
[131]	Forwarding local model updates to one-hop neighborhoods and suitable for large and dense D2D.	Single point of failure problem
[132], [133]	Replacing the central server with blockchain.	Privacy

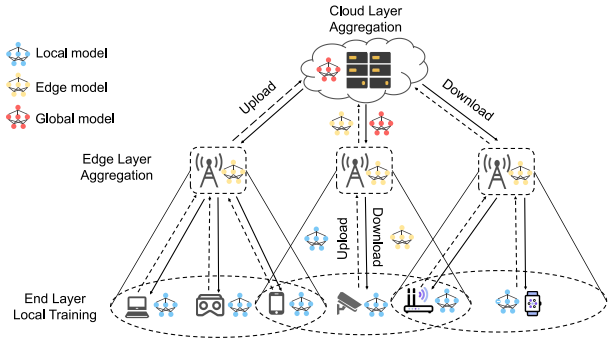


Fig. 11. The end-edge-cloud hierarchical FL architecture.

hierarchical FL involves multiple distributed heterogeneous devices (or participants) with different computation capabilities [123]. Given the multi-levels of model aggregation, how to design the communication strategy (e.g., convergence frequency) to achieve a tradeoff between model accuracy and communication efficiency requires further investigation.

Facing these challenges, some researchers study communication optimization strategies and perform convergence analysis. An intuitive way to improve communication efficiency is utilizing compression technologies such as sparsification, quantization, and sketching [124]. However, gradient compression may degrade the performance in terms of accuracy and convergence speed due to the weighted distance between the local distribution and global distribution. To address this issue, Mhaisen et al. [125] attempted to equalize class distributions among edge devices, by exploiting the advantages of both cloud and edge servers, i.e., frequent local aggregations at the edge servers and few aggregations in the cloud server [126], [146]. Specifically, for the high-cost global averaging, it is recommended to decrease the global averaging frequency and increase the local averaging frequency [127].

Determining aggregation frequency at edge and cloud levels in a three-layer FL system is also a crucial problem. There are two qualitative guidelines: (i) frequent local model averaging can reduce the number of local iterations, and (ii) making the edge data set IID distributed can reduce the communication overhead with the cloud [121]. Following [127], Wang et al. [128] proposed a cluster-based FL mechanism, which included intra-cluster communication in the same edge clusters and hierarchical aggregation between a lead edge node and a remote cloud parameter server. Deng et al. [129] also considered exploiting edge aggregation to avoid frequent communications with cloud servers. They proposed a method to solve the cost minimization problem in HFL. Another way to decrease communication costs is identifying irrelevant updates and precluding them from being uploaded for reduced network footprint. By avoiding uploading irrelevant updates to the server, it can reduce the communication overhead while guaranteeing the learning convergence [130].

Decentralized FL (DFL): Although CFL can preserve privacy without exporting data to third parties, it has the drawbacks of single-point failure and scaling issues [131]. Different from CFL, DFL does not require central update aggregations. DFL participants at all levels are connected in a peer-to-peer (P2P) architecture. In each iteration, each client performs training based on a local data set and then interacts with adjacent clients via a consensus-based manner, where all model parameters are encrypted. Due to the decentralized features, DFL can be integrated with P2P-based blockchain to build secure distributed systems. The blockchain network can replace the centralized model exchange and aggregation due to the distributed ledger technology [147]. Moreover, incentive mechanisms can motivate users to participate in collaborative training and upload model parameters [132]. To further improve the FL system using blockchain, the authors of [133] designed strategies to allow unsuspecting parties to participate in privacy-preserving learning and sharing gradients. However, due to the unique features of the blockchain, such as decentralized architecture, privacy concerns, node resource requirements, and multi-party consensus, the EECC architecture faces many challenges when integrating blockchains, such as trusted computing and transmission.

4) Few-Shot Federated Learning: Despite the existing efforts to reduce communication overheads, current FL approaches still use iterative optimization technologies, requiring many rounds of communication between the devices and the central servers to train a global model. Motivated by the advances of meta-learning and few-shot learning, iteration rounds can be further decreased. The authors of [148] proposed a promising few-round learning for FL, where an initial model was prepared to limit the number of model exchange rounds in FL to a fixed number R.

To further reduce communication rounds, one-shot FL becomes a solution that only needs a single round of communication between the clients and central server [149]. In one-shot FL, each client trains a local model and applies ensemble methods to capture global information across the device-specific models. Besides, the client-to-server communications can be controlled via specific protocols, such as setting thresholds based on local validation errors. The one-shot FL is commonly used in two settings, i.e., traditional supervised learning and semi-supervised learning. Recently, DOSFL [150] was proposed to combine data distillation and one-shot FL, where each client distills its data and uploads learned synthetic data, label, and learning rate to the server, instead of transmitting bulky gradients or weights. Furthermore, the authors of [151], [152] proposed Fusion Learning for hierarchical client-edge-cloud architecture, which trains a DL model by integrating both fusion and FL methods.

C. Optimization Technologies for Distributed Inference

1) Model Acceleration: Although model compression can facilitate DNN inference on mobile devices, current solutions to deploy DNNs on low-power and resource-constrained mobile or edge devices (e.g., smartphones) are still inefficient. To address the challenge, hardware-based optimizations are proposed to reduce DNN inference costs. Next, we introduce hardware acceleration, and software acceleration, along with software and hardware co-design acceleration.

- *Hardware Acceleration (HA):* HA is the underlying optimization mechanism that improves the model inference efficiency by fully utilizing the computing abilities of hardware. For most mobile devices, CPUs and GPUs are general-purpose processors for model acceleration. Recently, Google developed Tensor Processing Units (TPU) [153], a high-speed custom machine learning (ML) chip that is 15-30 times faster than standard CPUs and GPUs. Some other examples are HiSilicon neural processing unit (NPU) [154], Qualcomm Snapdragon Neural Processing Engine (SNPE), and field-programmable gate array (FPGA) [155]. Next, we introduce the related works about the CPU/GPU and the FPGA-based acceleration technologies.

CPUs/GPUs: Mobile CPUs and GPUs are the most common hardwares for on-device or edge inference, and each terminal device has different numbers of CPU and GPU cores. Therefore, making full use of heterogeneous hardware resources on the device can greatly reduce the delay and energy consumption of DL model inference. However, DL inference on mobile CPUs suffers from two issues: poor performance scalability and energy inefficiency, due to improper task partitioning, unbalanced task distribution, and unawareness of model behavior. To address these issues, the authors proposed AsyMo [156], with joint consideration of model structures and hardware asymmetric characteristics of the CPU. Compared with the CPU, the GPU has more power and better energy efficiency. Yet, mobile GPUs also face the inefficient issue resulting from the improper partition of computing workload. Jiang et al. [157] proposed a heuristics-based workload partitioning approach, considering both model performance and computation overheads on mobile devices. In addition, mobile GPUs suffer from significant kernel launch overhead [158]. To maximize the utilization of mobile GPUs, concurrent execution of heterogeneous operators [159] and exploiting activation sparsity [160] are effective methods. Leveraging the heterogeneous computing capabilities of CPUs and GPUs, real-time multi-person pose estimation and mobile action recognition could be deployed on mobile devices [161].

FPGA: The CPU-based and GPU-based solutions are widely utilized in on-device and edge inference. However, some state-of-the-art DNNs are designed to adopt network sparsity and compact data types, introducing irregular parallelism and custom DNN structure. Due to the redundant design of general-purpose processors [162], [163], it becomes difficult to execute DNN models on GPUs. This dilemma motivates the study of specialized accelerators tailored for neural networks. More recently, FPGA-based hardware is proved to be well-suited for DNN acceleration architecture due to its specific customizability. FPGA outperforms GPUs in throughput, parallelism, and energy efficiency in edge computing since it enables CNN acceleration with arbitrary convolution window sizes [164]. Researchers designed customized FPGA architecture for CNNs with different structures, including regular ConvNet [165], FPGA-based overlay processor for lightweight CNNs (e.g., MobileNet, ShffleNet) [166], FPGA-based accelerator for deconvolutional neural network.

- *Software Acceleration (SA):* Other than the hardware acceleration, some researches provide software-based acceleration strategies. For example, Lane et al. [167] presented DeepX, a software accelerator for DL. DeepX decomposes DNNs into various types of unit blocks and allocates them to local device processors (e.g., GPUs, LPUs, and CPUs). The authors also designed two resource control algorithms (i.e., Runtime Layer Compression and Deep Architecture Decomposition) to perform principled resource scaling on mobile platforms to maximize energy efficiency. In [168], the heterogeneous computing framework RenderScript and hardware on commodity android devices were combined to accelerate the execution of DL models. With the Renderscript, DeepAction [169] offloaded the image matcher execution to the GPU for acceleration.

- *Hardware/Software Co-design:* There are a variety of HA and SA technologies for accelerating DNN inference in mobile and edge devices. Nonetheless, it is hard to strike a balance between high model performance requirements and limited hardware resources relying on the one-sided design. Thus, for efficient DNN inference, it requires hardware and software co-design (Co-HW/SW) [170]. We summarize some related works in Table VII.

2) Model Partitioning: Uploading raw data to remote servers leads to huge communication overhead and becomes the bottleneck for DL inference. To reduce end-to-end latency and improve energy efficiency, the DL models can be segmented into multiple partitions and then allocated to distributed computation devices, e.g., local CPUs and GPUs, edge nodes, and cloud servers, as illustrated in Fig. 12. The model partitioning includes three types: partition between servers

TABLE VII
HARDWARE/SOFTWARE CO-DESIGN

Framework	Hardware	Software	Model
SECDA [171]	FPGA	SystemC Simulation	CNN
TT-YOLOv5 [172]	FPGA	Tensor Train Decomposition	CNN
HSCoNAS [173]	CPU, GPU	Neural Architecture Search	CNN
Tiny-YOLOv3 [174]	FPGA	High Level Synthesis Tools	CNN
FantastIC4 [175]	FPGA	Entropy-constrained Training	MLP

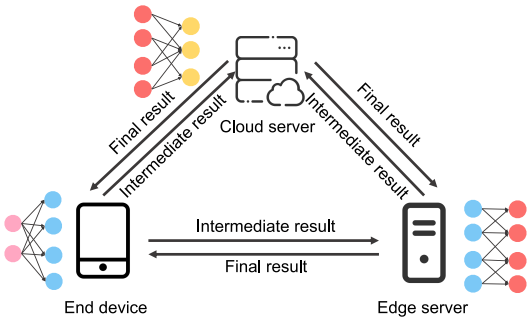


Fig. 12. Illustration for model partitioning in EECC.

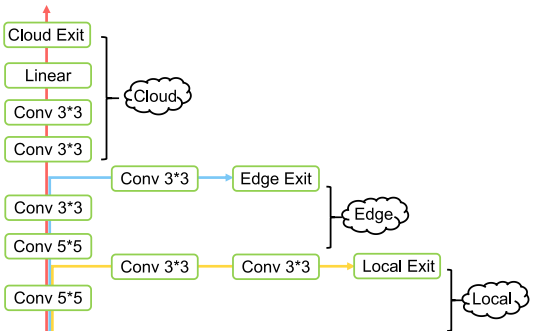


Fig. 13. Illustration for model early exiting in EECC.

and devices [57], [176], [177], the partition between devices horizontally [178], [179] and vertically [180].

The challenge of model partitioning lies in how to select optimal partition points of neural network layers. To address this challenge, several key factors must be taken into consideration: i) the resource capacity of the edge device; and ii) the size of the intermediate data. Generally, partitioning at lower layers may generate larger sizes of intermediate results, leading to increased transmission latency. Wang et al. [181] searched for model partitioning strategies based on reinforcement learning to adapt to resource capabilities of edge platforms. To reduce the size of the transmitted intermediate data while guaranteeing the model accuracy, knowledge distillation can be utilized to compress the subsets of DNN run on the edge [182]. Besides, the model partitioning decision may be affected by wireless channel conditions. In practice, it is hard to predict the channel state information before model partitioning.

For end-edge-cloud co-inference, it is not necessary to download all the DNN layers. Banitalebi-Dehkordi et al. [183] studied a joint optimization problem that included optimal model placement and online model splitting for device-edge co-inference. In [184], the optimal DNN partition was formulated as a min-cut problem in a directed acyclic graph and the authors proposed an execution latency measurement method to estimate the inference latency. Besides, by exploiting the strong sparsity and fault-tolerant characteristics of intermediate features in DNN, BottleNet++ [185] was proposed to achieve a high compression ratio. Recently, the model partitioning technology has been applied in the industry products for collaborative cloud-edge intelligent services [183].

3) Model Caching: Model caching is used to accelerate model inference, i.e., achieving a good trade-off between latency and prediction accuracy by caching the pre-trained low-complexity models on end devices and high-complexity models on edge or cloud servers. Instead of splitting models, model caching relies on full models and decides caching decisions at run-time based on computation capability of local, edge, and cloud computing nodes and latency requirements [186], [187]. However, how to effectively manage and utilize deep learning models at each edge location to provide inference performance guarantees is a challenge. EdgeServe [188] was proposed to adopt the term caching for efficient DL model caching at the edge.

4) Model Early Exiting: Model early exiting provides a fast solution to satisfy diverse requirements of inference by mapping sections of a DNN onto a distributed computing hierarchy with multiple output nodes [189], [190], as shown in Fig. 13. As long as the target accuracy threshold is met, the model on the local device or edge device will return the results, otherwise, the edge offloads the remaining inference tasks to the cloud. With joint inference, model early exiting minimizes communication and resource usage.

To increase the estimated accuracy and recognition robustness on the edge, authors of [191] introduced expert side branches trained on a particular network. Combining the early-exiting mechanism and model partitioning can significantly accelerate end-edge DNN co-inference while achieving high accuracy [192], [193]. The integration of early exiting and intermediate feature compression can also efficiently accelerate end-edge co-inference [194]. Nevertheless, choosing the optimal exit point is challenging as the delay and accuracy of each point cannot be known in advance. Ju et al. [195] used multi-armed bandits to learn the optimal exit point for mobile-based DNN inference. Wang et al. [196] designed the scheduling early exit algorithm based on dynamic programming to make the optimal plan with polynomial computational complexity. Unlike existing efforts on offline searching, Autodidactic Neurosurgeon [197] is based on a contextual bandit learning algorithm that can automatically learn the optimal partition point on the fly. However, most efforts have focused on CNNs (e.g., object detection [198], visual object tracker [192]), early exiting for other models (e.g., RNN, GAN, seq2seq, and VAE) remains to be explored.

5) Model Multi-Tenancy: With the popularity of DL-based applications, mobile devices such as smartphones, drones, intelligent driving cars, and augmented-reality headsets may run more than one DNN model simultaneously. For example, augmented-reality headsets need to concurrently run DNN

models for object detection, voice recognition, and action recognition. These models compete for limited resources. As a result, we need a model multi-tenancy technique to dynamically allocate resources for those concurrent applications. Generally, AI multi-tenancy can be achieved by utilizing concurrent model executions (CME) and dynamic model placements (DMP) [199]. DMP allows the deployment of DL models on GPUs and TPU resources of edge devices to achieve AI multi-tenancy. CME supports the concurrent execution of multiple DL models to improve the overall efficiency of DL inference. NestDNN [200] was a resource-aware multi-tenant on-device DL framework for continuous mobile vision. It considers the dynamics of run-time resources and provides a flexible resource-accuracy trade-off for each DL model to adapt model resource requirements to the available run-time resources. HiveMind [201] was designed for optimizing multi-model concurrent workloads, performing the optimized directed acyclic graph on the GPU, and trying to run concurrent workloads.

Different from NestDNN and HiveMind, MainStream [202] was proposed to dynamically tune degrees of work-sharing among concurrent applications. This sharing mechanism is achieved by transferring learning from a common DNN model, thus decreasing computing time. Furthermore, due to limited runtime memory for wearable devices and edge devices, DeepEye [203] used an inference software pipeline that interleaves the execution of computation-sensitive convolutional layers with the fully-connected layers. Beyond this, the memory can be further minimized by a memory caching scheme and model compression. In addition, the authors performed experiments on a Kubernetes cluster to demonstrate the feasibility of an edge cloud with multiple tenants and hosting providers [204]. AI multi-tenancy technologies open up new opportunities for flexible deployment of multiple DL services on edge devices and AI accelerators.

D. Summary and Lessons Learned

In this section, we have reviewed several main optimization technologies for distributed AI training and inference in EECC. There are three lessons learned.

1) Metric-Aware Optimization Strategy Selection: For a specific intelligent service in the end-edge-cloud distributed AI, there are usually multiple technologies to optimize the service. However, it may be difficult for us to choose the perfect optimization technology for each service quality. Therefore, we need to quantitatively analyze the factors affecting service performance and resource constraints, and explore the trade-offs between different metrics to help improve the efficiency of intelligent service deployment.

2) Coordination of Different Optimization Strategy: These optimization technologies are often used independently to enable end-edge-cloud collaboration. However, most optimization strategies can optimize only one single metric. For some complex tasks, e.g., Metaverse and autonomous driving, coordination of different optimization strategies should be thought over to improve multiple metrics concurrently.

3) Preferred Performance Metrics: Computation and communication overheads have become essential metrics for evaluating the utility of distributed AI. With increasing model depth, the model accuracy is no longer an intractable problem of distributed AI, instead, the increasing computing and communication overhead is imposing a heavy burden on the resources-constrained edge devices. As a result, most existing works for distributed training are exploring how to speed up training and reduce communication costs. In addition, the rise of FL has made privacy an important indicator, as it is designed to break down data isolation while protecting data privacy. It can also be combined with splitting computing or gradient compression to reduce computation latency and communication overhead. For example, end devices in FL with weak computing capability can split the model and deploy part of the model on edge devices to reduce the computing burden. End devices can also use gradient compression to improve communication efficiency by aggregating gradients.

IV. Security and Privacy in DAI-EECC

The EECC architecture can take advantage of distributed hierarchical structure to provide various computing services. In this section, we firstly summarize the challenges in DAI-EECC architecture from four aspects, i.e., data storage, computing, transmission, and system management.

A. Key Challenges

1) Data Storage: In the EECC architecture, data is generated and stored in end devices and edge servers. Compared to the powerful cloud data centers, the end layer and edge layer have limited defense capability against attacks. In addition, EECC architecture faces the device stability issue. Specifically, edge nodes are susceptible to breakdown due to uncertain external factors, resulting in data loss or tampering. Although cloud services are usually securely maintained by third-party service providers, unreliable cloud services may also exist due to data poisoning. Therefore, guaranteeing the integrity and availability of data at the cloud, edge, and end layer is a challenge.

2) Computing: From the vertical perspective, devices at different levels under the EECC architecture have their particular advantages, such as the location proximity benefit at the edge layer and the elastic computing at the cloud layer. From a horizontal perspective, the EECC system can provide efficient computing services with multi-scene collaboration. However, as the devices at all levels belong to different stakeholders, the potential of collaborative computing cannot be fully exploited due to distrust issues. Therefore, how to construct a trusted computing framework to provide reliable and secure services is an urgent problem to be solved.

3) Transmission: Compared to centralized CC, the EECC service is provided by the cooperation of many nodes, and data is distributed on different nodes, forming data islands. Therefore, multi-party data sharing requires secure access control mechanisms to ensure privacy and security. However, due to various attacks in the complex network environment, the security of data transmission cannot be guaranteed. In

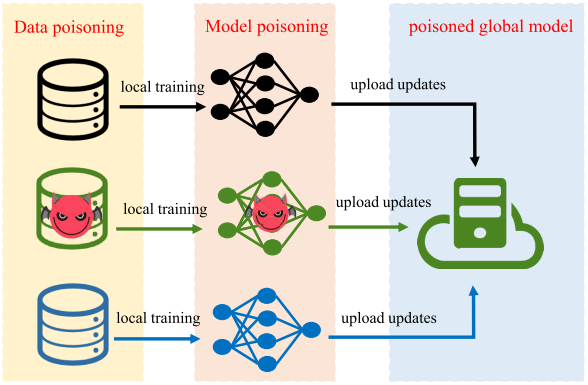


Fig. 14. Illustration of data poisoning and model poisoning.

response to this, it is necessary to explore a safe and reliable data transmission strategy, and design incentive mechanisms to encourage data sharing among nodes, forming a trusted and active data-sharing environment.

4) System Management: The EECC architecture is an open system in which heterogeneous end devices are connected to edge servers and cloud data centers. This unaudited device access brings uncertainty and security risks to the system. Therefore, device management in edge environments (such as dynamic access, departure, and identity identification of IoT devices) requires a secure management and audit approach. Traditional centralized management architecture inevitably faces the risk of central node failure. Therefore, designing safe and reliable system supervision and equipment audit mechanisms is worth studying.

Next, we will introduce in detail the threat sources that lead to the above-mentioned security and privacy challenges and the corresponding countermeasures.

B. Security Threats

The security threats include unauthorized access, maliciously manipulating the global model, or reducing the model performance. The distributed scenarios have higher security risks than centralized learning because a large amount of distributed end devices with limited computing and storage resources are more vulnerable to attacks. Moreover, the global model is a white box to the compromised devices, who can observe the model structure, learning algorithm, hyper-parameters, and so on. Thus, it is easy for the attackers to manipulate the performance of the model via modifying the training data or model parameters. The security threats exist in both storage, computing, transmission, and management phases. We introduce the typical attack methods for each phase, respectively.

1) Data Security: Data poisoning is that attackers inject malicious data into the training dataset to make the target model less effective [205], [250] (e.g., label-flipping). As shown in Fig. 14, the attacker replaces the target label with an incorrect one. In distributed training (e.g., FL), attackers can collude with each other and inject malicious samples into their local models, resulting in great catastrophe in the global model aggregation at the edge or in the cloud [251]. In addition, the attacker can also generate fake samples similar to the target data through the GAN [252].

2) Model Security: Unlike data poisoning, model poisoning directly changes the local model parameters or gradient update, making the global model deviate from the correct direction to the greatest extent [208], [215]. Model poisoning can be divided into untargeted attacks and targeted attacks.

- *Untargeted Attacks:* The untargeted attack is similar to the Byzantine attack, in which the malicious workers destroy the global model by arbitrarily modifying data or model parameters.

- *Targeted Attacks:* This type of attack is also known as backdoor attack, in which the attacker signs targeted labels to the samples that have some specific attributions or features, and then trains a backdoor model with these backdoor data sets [214], [216]. A trained backdoor model can correctly predict samples, but once encountering the trigger data, it will output wrong results, e.g., identifying all the green cars as birds [216].

3) Network Security: Distributed AI relies on the network layer to connect a large number of terminal devices and edge devices. As the increasing number of devices, the scale and number of networks also increase, and the security problem becomes more serious. The common network attacks include physical attacks [218], distributed denial of service (DDoS) [219], and Man-in-the-Middle [220]. In the distributed environment, these attacks are easier to achieve since a large number of devices have dense geographical distribution and limited computing resources. For example, in the computation offload tasks, an attacker can offload the data packets containing malicious codes to other terminals. As a result, the receiver cannot provide computing services and continue to send the data packets to other devices, eventually resulting in the whole network breakdown.

4) Access Security: In the traditional computing environment, data is stored in a central server, user information and access services are within the trusted boundaries of the enterprise. However, in a distributed architecture, data is stored on a large number of decentralized nodes. The external adversaries can easily usurp their legal identities by phishing, password guessing, or cross-site scripting attacks. Moreover, the adversary can also penetrate the central server to tamper with the access control mechanism. In the centralized distributed AI system, once the central server is compromised, the system cannot prevent illegal access. Thus, the system must have a trustworthy access control mechanism that is fault-tolerant and tamper-proof to guarantee the confidentially, integrity, and availability of data [223].

C. Security Defense Technologies

We introduce several security defense technologies to tackle data poisoning, model poisoning, and untrusted threats.

1) Defenses for Data Poisoning: To defend against data poisoning, Xie et al. [253] focused on the defense in local SGD, i.e., robust aggregation and moving averaging. Robust

aggregation uses trimming averages to filter out the poisoned model. The moving averaging reduces the extra variance/error caused by robust aggregation and data poisoning. Besides, Fung et al. For the GAN-based data poisoning, Zhang et al. [252] found that pivotal training is a potential defense in the FL setting. This method distinguishes the received local gradients by creating an extra predictive model to infer which participant a certain gradient belongs to, thus preventing the discriminating behavior and eliminating the influence of poisoned local updates.

2) Defenses for Model Poisoning: Robust aggregation is the most effective method to resist model poisoning attacks. For the untargeted poisoning attacks, several Byzantine-robust aggregation algorithms methods have been proposed, such as Krum [209], Bulyan [211], and Trimmed-mean [210]. They eliminated incorrect local updates caused by Byzantine errors by minimizing the Euclidean distances between selected local models. However, these methods are less effective when a large number of Byzantine adversaries exist in the system.

For the targeted poisoning attacks, some behavior-based aggregation methods are proposed, which remove the potential malicious gradients and average the remaining subset by measuring the behavioral difference between honest and malicious workers. For example, Muñoz-González et al. [213] proposed adaptive federated averaging (AFA), which calculated cosine-similarities between local gradients and discarded the gradients whose statistical characteristics of similarities (i.e., mean, median, and standard deviation) were out of range. Similarly, Fang et al. [207] computed a score for each gradient and remove the gradients with the lowest scores. However, Awan et al. [212] found that these methods had poor performances on non-IID data since the gradients from different distributions of honest workers were quite different. To solve this problem, they extended the AFA in a non-IID setting. Specifically, they found that the pairwise cosine similarity between any two malicious clients is always smaller than that between a malicious and an honest client and that between any two honest clients. Namely, if two clients have high-level similarities, they are suspicious to be malicious. In this case, they will reduce the learning rate and the probability of being selected of the suspicious clients. Besides the robust aggregated-based methods, norm clipping and weak differential privacy [215] and pruning and fine-tuning [217] can also mitigate or even eliminate the poisoning attacks.

3) Defenses for Cyber Attack: Traditional defenses against cyber attacks include encryption, authentication, access control, and so on. With the development of ML, there is also a lot of research using AI to protect networks from attacks. These methods mainly detect abnormal behavior that an attack is possibly taking place. The authors of [221] used a CNN model, an RNN model, and a hybrid CNN+LSTM model to detect the cyber attack, respectively, and the highest detection accuracy on the CICIDS2017 dataset is 98.16%. However, these methods are passive defenses that cannot actively look for attackers. One of the active defense mechanisms is deception-based defense, which creates deception and some continuous unpredictable changes in network configures to perplex and confuse the attackers, or to lure them toward some pre-deployed honeypot traps [222].

4) Blockchain Technologies: Recently, blockchain technologies are used to improve the defense level in EECC architecture. Blockchain has several advantages: (i) data is unalterable, traceable, and jointly maintained by multiple parties. Thus, the integration of blockchain into EECC architecture can ensure the data integrity and availability in the process of collaborative computing [254]. (ii) The smart contracts make the blockchain-based EECC system a secure and trusted computing service platform, providing efficient and secure computing services. (iii) As a platform for data interaction and token transfer, blockchain can realize incentive mechanisms through tokens to promote data sharing between multiple scenarios and encourage multi-party collaborations. (iiii) The data transparency and auditing feature of blockchain can be used to manage device access and improve system security [224], [225], [226].

D. Privacy Threats

The end devices in the EECC framework will generate massive data, which often contains a large amount of private information. Although the original data is stored locally and the end device or edge device only needs to upload the processed data, the intermediate data still faces the risk of privacy disclosure. For the end devices, they may regard personal training data, attributes, and member information as private information. For the cloud or edge, the trained model is private information, because the training process usually has a massive expense, and the leakage of model parameters to opponents will lead to serious economic losses. Privacy threats occur when an attacker is able to link a record owner to a sensitive attribute in a published data table. In the following, we introduce several privacy threats.

1) Membership Inference: Given a training sample, a membership inference attack is to judge whether a training sample is involved in the training process. If an instance is inferred as the training data, the data owner's privacy may be leaked. For instance, a patient whose clinical records are used to train a heart disease prediction model is likely to have heart disease. The vulnerability of this type of attack stems from the differences of output distributions of the model to membership and non-membership inputs. The differences are even greater when models overfit the training data.

Membership inference attacks can be divided into black-box and white-box inferences. Black-box inference attacks usually assume that the attacker can only obtain the output probability vector of the model. Shokri et al. [255] firstly proposed this attack, where a mass of samples from the training data set and non-training data set are constructed, respectively. Then, they used the samples from the training data set to train the shadow models to imitate the performance of the target black-box model. After that, they used the outputs of shadow models from the training data set and non-training data set to train a binary classifier to distinguish the membership and non-membership. Considering the lack of target training data in practice, Zhang et al. [227] used GAN to generate samples

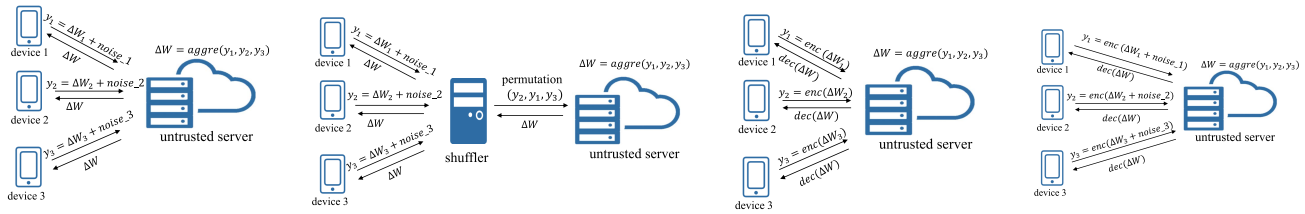


Fig. 15. Several privacy-preserving technologies.

that have the same distribution as the original data records to fill the gap in attack data.

The other inference attack assumes the white-box setting, which means that the attacker can obtain the parameters and gradients of the middle layer of the model. Membership inference attacks in distributed settings can be naturally treated as white-box, because both the participant, edge, and cloud can obtain the model updates. Although the white-box settings are more transparent than the black-box settings, the authors [242] found that it is not effective to apply the inference on the output layer under the black-box setting to all activation functions of the white-box model. Thus, they proposed the gradient ascent attack, where the attacker runs gradient ascent on a target data sample and observes whether its increased loss can be drastically reduced in the next communication round, if so, the sample is very likely to be in the training set.

2) Attribute Inference: In addition to the leakage of membership information, Lyu and Chen [243] demonstrated that the shared epoch-averaged local model gradients can also reveal sensitive attributes of local training data of any victim participants. They assumed that the malicious server can infer the sensitive binary or categorical demographic-related attributes. The malicious server reconstructs the sensitive attributions by matching the original gradient updated by the victim participant and the virtual gradient calculated on a virtual input. If the distance between the two gradients is small enough, even though the attacker did not know the attribution in the original input, he/she can still obtain the sensitive attributions from the virtual input. Considering the model updates may be anonymous and shuffled before aggregation to obscure the source, the attribute attack can also be coupled with a deanonymization attack to enhance the attack effect. The authors of [244] utilized the bias error signal among different model updates caused by the non-IID data distribution to re-identify the corresponding participants, and showed that the deanonymization can serve as a stepping stone for attribute inference to improve attack performances and further associate recovered attributes with identities.

3) Reconstruction Attack: Reconstruction attack is referred to as reconstructing the training samples of the target device from the aggregated model. Actually, the model parameters can be viewed as encoded information from training samples. Therefore, if a corresponding decoder could be constructed, the private data or statistics can be recovered inversely. Because GAN model can generate data that has the same distribution as real data, the recent works often take GAN as the decoder to recover training data. Hitaj et al. [246] utilized the shared model as the discriminator to train a GAN and train a generator model to successfully mimic the training samples. To reduce the intrusion of attacks on the training process, Wang et al. [245] assumed the server was malicious, who used a multi-task GAN to generate training inputs. The multi-task discriminator restrained the training of the generator, as a result, the generated samples were targeted to a specific client and compromised the client-level privacy.

4) Model Extraction: In the model extraction attack [248], [256], an adversary uses a generative model to generate input data and then queries the target model to get outputs. With the inputs and outputs, it can train a clone model. But this attack is not feasible since it needs a lot of queries, which is easy to be detected by anomaly detection. Considering this challenge, the authors of [247] proposed a data-free model extraction attack. The adversary uses the explanations for computing the exact gradients to train the generative model and reduces the number of queries. However, it cannot steal the complete model. Zhu et al. [249] proposed Hermes attack, which fully stole the whole victim model, including the hyper-parameters, parameters, and architecture through accessing the PCIe bus between the host machine and the GPU-based devices.

E. Privacy Protection Technologies

The above privacy threats share an underlying property, i.e., the attacker infers privacy information from the model parameters or updates. As a result, the defenses for model parameters and updates are universal for all protection technologies. Instead of displaying the corresponding defenses for each privacy threat, we highlight the prominent defense methods: differential privacy and encryption technologies, which are shown in Fig. 15.

1) Differential Privacy (DP): DP has become an effective way to protect private data in distributed learning [228], [229]. The basic concept of DP is to inject Gaussian or Laplace noise into the data to guarantee that the adversary cannot infer the original data from the noisy output with high confidence. In the DP-based distributed learning setting, the central servers are assumed to be trustful and can inject noise into the aggregated gradients to defend against the outer adversary. The privacy level is controlled by the privacy budget. A larger budget means a higher privacy loss, which would be accumulated in each training epoch. To bound the total privacy loss in the training process, some advanced composition theorems

TABLE VIII
A SUMMARY OF SECURITY AND PRIVACY THREATS IN DISTRIBUTED LEARNING AND COUNTERMEASURES

Threats	Attack type	Attack methods	Countermeasures
Security	Data security	Label flipping [205]	Anomaly detection [205]
		Fake data generation [205]	Robust losses [206]
	Model security	Untargeted attacks [207], [208]	Robust aggregation [209], [210], [211]
			Remove potential malicious updates [212], [213]
		Targeted attacks [214], [215], [216]	Norm clipping [215]
			Differential privacy [215]
			Pruning and fine-tuning [217]
	Network security	physical attacks [218], DDoS [219], Man-in-the-Middle [220]	AI detecting [221]
			Deception based defense [222]
	Access security	Usurp legal identity [223]	Blockchain [224], [225], [226]
Privacy	Membership inference	GAN-based attack model [227]	DP [228], [229], [230], LDP [231], [232]
		Gradient ascent [242]	LDP+shuffling[233], [234]
	Attribute inference	Matching target gradients and virtual gradients [243], [244]	Encryption [235], [236]
	Reconstruction attack	GAN-based inputs reconstruction [245], [246]	Encryption+LDP [237], [238]
	Model extraction	Imitation training [247], [248]	Compression+LDP [239], [240]
		Accessing to PCIe bus [249]	Compression+ Encryption [241]

were proposed to track the cumulative privacy loss [257]. The training would stop when the loss reached a threshold. This method protects the user-level privacy, i.e., an outer adversary cannot know whether a user participates in the training or not from the aggregated model.

However, the central server may be honest-but-curious, and it can infer the privacy of participants from the uploaded gradients. To defend the malicious server, some works proposed the local differential privacy (LDP) [229], [232], [258], which applied DP at the local gradients before aggregating. LDP protects record-level privacy, i.e., an adversary cannot know whether a particular sample participant is in the training or not. LDP has a stronger privacy guarantee than DP, but it also suffers from huge utility degradation, especially with a large number of participants. To fill the gap, the shuffling technique was combined with DP [233], [234], [259]. It anonymously shuffles the local gradients before aggregation to enhance privacy. However, anonymization may lead to malicious users tampering with data. Furthermore, it is difficult for the server to evaluate the participants' contribution to calculate the payoff.

2) Encryption Technologies: Secure Multi-Party Computation (MPC) [260] is the most commonly used method to protect data privacy in distributed scenarios involving multi-party computing. It enables multiple participants to securely compute a public function without revealing their private data. Homomorphic Encryption (HE) is often used as the encryption technique in MPC, which allows performing addition or multiplication operations on a ciphertext, the decryption of which corresponds to algebraic operations on the plaintext [235]. The authors in [236] utilized an additively HE scheme to securely calculate the aggregated model updates. As a result, it prevented data leakage of clients at the central server. However, it is not safe enough if a client colludes with the aggregator as all the clients share the same secret key.

To solve this problem, authors of [261], [262] considered multi-key scenarios, where each client has a public-private key pair. The multi-key setting can not only defend against collusion attacks but also increase the flexibility and scalability of the system, as the client can join or drop out at any time. Although HE can defend malicious aggregators, it cannot prevent the malicious participant from launching inference attacks to obtain private information from the participant. Thus, some works [237], [238], [263] combined DP with encryption technologies to address such concerns, in which each participant injected noise into the local update before encryption. Encryption technologies usually have high communication and computation costs, which make them not applicable to large-scale distributed scenarios and resource-limited devices. How to trade off between efficiency and privacy is still an intractable problem.

3) Network Compression: Compressing network information can not only improve transmission efficiency, but also reduce the information exposed to the adversary. Both the gradient sparsification and gradient quantization mentioned above belong to the compression technology. However, the compressed gradients will still reveal sensitive information as they keep their own form [264], thus compression is also often combined with differential privacy or encryption to increase the level of protection. Li et al. [239] proposed a sketch-based framework for distributed learning, which transmitted messages via sketches to simultaneously achieve communication efficiency and provable privacy benefits. Wang et al. [240] used an auto-encoder to filter sensitive information and decrease the feature dimension, and DP noise is injected into the processed features to provide stronger protection. The authors of [241] proposed an end-to-end encrypted neural network, which locally encoded the gradient to a lower-dimension space, and decoded the aggregated gradients as a whole to defend against the untrusted central server.

F. Summary and Lessons Learned

In this section, we have discussed two issues, i.e., security and privacy. The typical attack methods and defense technologies are summarized in Table VIII. Privacy threats mainly exist in the process of model training and inference. Three key lessons learned from the security and privacy defense technologies are as follows.

1) Systematic and Comprehensive Security Framework: A systematic and comprehensive security framework is required

for the distributed AI empowered by EECC. AI technologies are vulnerable to attacks on data and models, like data poisoning and model poisoning. The distributed AI empowered by EECC not only inherits these defects but also faces new challenges due to the distributed architecture, such as dealing with network attacks, security authentication, and access control over a large number of devices and data. However, the existing researches generally focused on a certain threat, lacking systematic and comprehensive defense measures.

2) Proper Privacy Technology Selection: Privacy threats mainly exist in the process of model training and inference. DP and encryption are two main privacy defense technologies, which have achieved great success in reality. DP protects privacy by injecting noise. It is lightweight and easy to be integrated into the system but may degrade the model's performance. Compared to DP, encryption technologies have a higher level of privacy protection, but the high computation cost makes them hard to be deployed on resource-limited end devices. Since each privacy technology has pros and cons, proper privacy technology section should be conducted for different applications based on their requirements.

3) Laws and Regulations in Security and Privacy: The development of laws and regulations is the key to resolving the reality of privacy breaches. There are many privacy protection technologies, but privacy breaches still happen almost every day. This is because users have little control over the type and amount of data the company collects. Motivated by profit, organizations that collect data may maliciously exploit these data to target a specific object. Laws and regulations are effective ways to regulate them to comply with the privacy principles.

V. Applications in DAI-EECC

Nowadays, the EECC paradigm continues to have a profound impact on a wide range of intelligent applications, improving the quality of daily life and economic growth. In this section, several application scenarios are presented.

A. Smart Industry and Manufacturing

The smart industry and intelligent manufacturing refer to the integration of intelligence into the manufacturing processes, aiming to promote the automation of industrial manufacturing operations, e.g., automatic control, detection, monitoring, and prediction, as shown in Fig. 16. In the era of Industry 4.0, data generated from heterogeneous smart terminals (e.g., sensors and cameras) is growing exponentially. Traditional CC architecture faces several shortcomings, such as high latency and bandwidth along with low availability. With the support of EECC architecture, various requirements of intelligent industrial applications could be efficiently satisfied with effective collaboration among suppliers, manufacturers, and industrial customers. Next, we introduce several practical tasks in the EECC-enabled smart industry.

1) Control and Monitoring: In traditional cloud-based manufacturing, there are many control and monitoring equipments directly accessing to the cloud and generating massive industrial data. However, data transmission brings huge

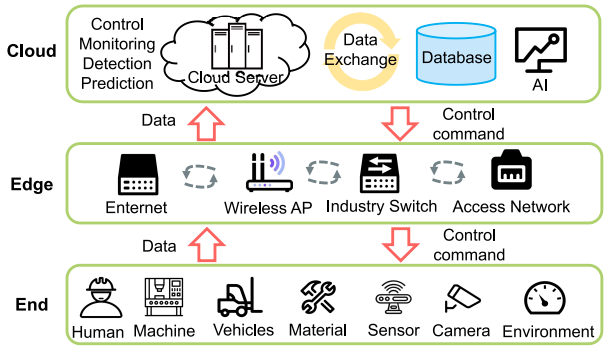


Fig. 16. An illustration of smart industry.

network pressure and unbearable latency, along with privacy risks. Guo et al. [265] proposed a cloud-edge collaboration architecture for 3D printers based on digital twin. By deploying time-sensitive services at the edge, real-time control and monitoring based on digital twin for additive manufacturing can be realized.

However, few industrial platforms consider the real-time latency demand of control, which severely hinders the deployment of delay-sensitive services. IndustEdge [267] was an extensible EECC orchestration component that can reduce the application latency via deploying the microservices of applications onto edge/cloud devices. Besides, task scheduling and resource allocation are still intractable problems in the control process, and the existing scheduling methods are difficult to keep a balance between algorithm complexity and performance. Thus, a cloud-edge-based two-level hybrid scheduling learning model was put forward for fast prediction of the cloud-edge collaborative scheduling results [268]. In terms of resource allocation, Zhang et al. [297] designed an EECC-based architecture to coordinate DNN inference tasks for the industrial IoT with limited resources, and placed part of the training model at the end and edge for pre-inference, maximizing inference accuracy.

2) Prediction and Detection: AI-enabled prediction and detection are crucial components of industrial systems to identify faults, malfunctions, or the effects of bad maintenance, and forecast future status in the manufacturing process. However, how to ensure low-latency and high accuracy requirements are critical challenges. EECC architecture can help solve these problems by scheduling heterogeneous resources. For prediction applications, Ding et al. [269] designed an LSTM-based load forecasting method deployed in the distributed cloud-edge environment. To reduce the transmission power consumption and delay, Wang et al. [270] exploited the differentiated capabilities of heterogeneous devices and designed bidirectional prediction-based underwater data collection protocols for the EECC system.

In terms of fault detection, through the cloud-edge collaboration framework in the power IoT, the author in [271] investigated imaged-based fault detection methods in power industry IoT and electric sites with the resource orchestration of edge server and cloud center. Besides, fault diagnosis is of great significance for timely detection of safety hazards of

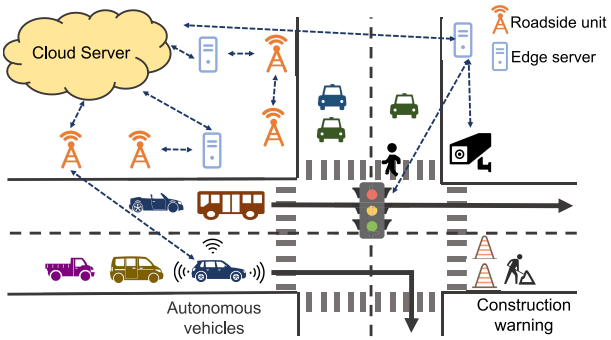


Fig. 17. An illustration of smart transportation.

machinery and ensures the normal operation of production. To address the problem of low accuracy and poor robustness of mechanical fault detection, Lu et al. [272] proposed an EECC intelligent fault diagnosis method based on the LSTM-VAE hybrid model.

B. Intelligent Transportation

The modern intelligent transportation system (ITS) relies on various remote sensors (e.g., road cameras, traffic density, and speed sensors) to sense road conditions and make scheduling decisions, as shown in Fig. 17. By introducing EECC into ITS, some services with ultra-low latency such as advanced IoV (Internet of Vehicles), ADAS (autonomous driving assistance systems), and real-time vehicle monitoring and scheduling can be better achieved [298], [299], [300].

1) Autonomous Driving: To ensure safe driving, the ADAS for autonomous vehicles integrates visual computing, radar monitoring, and a global positioning system to accurately detect nearby obstacles and make real-time decisions. Typically, the core functions of ADAS (e.g., perception, annotation, visualization, and navigation) are computation-intensive [301]. The onboard computing capabilities of vehicles are insufficient to perform these tasks and thus the support of a powerful cloud server is required. However, an autonomous driving car produces 4000 TB of data per day and needs to receive surrounding information and make real-time responses, which makes it insufficient to rely only on the cloud [302]. Therefore, this requires the synergy of edge computing and cloud computing.

In [273], EdgeDrive orchestrated storage, computing, and network components, which can support low-latency ADAS applications (such as smart navigation and weather notification) with an average system latency of less than 100 *ms*. Furthermore, some works are devoted to safe driving optimization, such as human abnormal driving behavior detection. Xun et al. [274] also presented a driving behavior evaluation mechanism based on vehicle-edge-cloud architecture. When vehicles run on the road, the driver behaviors are transmitted to the edge networks via telematics box, and the edge devices use the evaluation model trained by the cloud server to return results to vehicles. The cloud server continuously updates the model and regularly transmits it to the edge side [303].

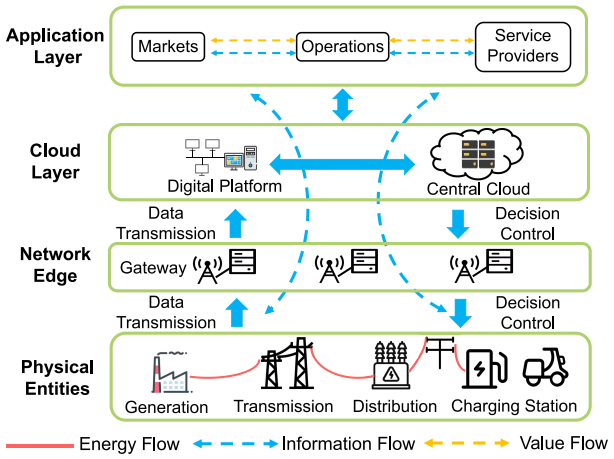


Fig. 18. An illustration of smart grid.

2) Public Vehicle Monitoring and Scheduling: Public vehicle monitoring and scheduling play important roles in public transport management. For example, the real-time monitoring of urban road traffic flow can provide intelligent scheduling and relieve traffic jams. In general, the network edge can transmit all the captured surveillance videos to the cloud server. However, the increasing traffic monitoring data volume has brought great pressure to the cloud storage and processing. To address this issue, the vehicle video detection model can be firstly trained in the cloud and deployed on an edge-computing-based video analytics system for real-time traffic monitoring [275], [276]. Moreover, EECC can help public vehicle (PV) systems improve traffic efficiency. ECPV [277] was proposed to improve vehicle occupancy ratios by scheduling ridesharing among travelers. However, the resources in the ITSs are complicatedly coupled, and the orchestration of multiple resources is hard to be solved in a real-time manner. In [278], Yuan et al. explored cross-domain resource scheduling and orchestration for the smart road, and utilized swarm intelligence to jointly optimize the information flow for the PVs, which enables agents to learn cross-domain optimization policies.

C. Smart Grid

Smart grid is the electricity supply network that uses digital communication technologies to achieve reliable and efficient power usage, usually relying on cloud-based infrastructure. With the rising number of smart meters, there are some limitations on cloud computing. For instance, processing sensor data in the cloud is inefficient for power supply/demand prediction due to the significant latency of data transmission. By bringing computations closer to IoTs and the data sources, EECC-based smart grid architecture, as shown in Fig. 18, can fully exploit the distributed resources, reduce communication latency, and solve practical problems in power IoT. Examples include network congestion alleviation [304], smart meter connectivity verification [279], equipment surveillance [280], and high impedance faults detection [305].

Recently, in consideration of environmentally friendly and energy-saving emission reduction, the number of electric

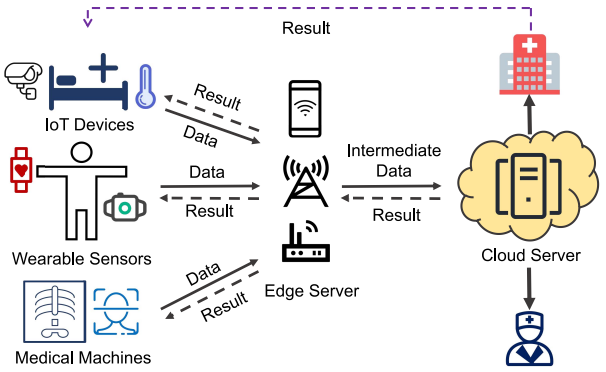


Fig. 19. An illustration of smart healthcare.

vehicles (EVs) has increased rapidly. It brings some challenges to the safe operation of the smart grid network. Zhong and Xiong [281] proposed an ordered charging scheduling method for EV charging in the cloud-edge collaborative environment. The power supply plans are designed according to the status of edge nodes (e.g., charging stations and renewable plants, etc.) and cloud side. Besides, the operation efficiency of the power grid is a critical factor for grid stabilization. V2G (Vehicle 2 Grid)-enabled EVs can improve this situation via load matching and current harmonic filtering. The authors of [282] proposed a smart framework based on IoT and edge computing to manage V2G operations efficiently, aiming to create an optimal charging schedule for each EV and increase grid reliability.

To improve the management of EV charging stations, Zhang et al. [283] utilized reinforcement learning at the network edge to decide plug-in EV charging pricing strategies, where service providers can quickly respond to the dynamic needs of users and load of grid. Moreover, data privacy and cost issues also attract attention. A decentralized scheduling method [306] was designed for EV charging and discharge management based on cloud-edge collaboration, ensuring the resource-efficiency and privacy of the energy supply plan.

D. Smart Healthcare

Benefiting from the rapid development of smart wearable devices, biosensors, and AI technologies, smart healthcare has been widely promoted to facilitate traditional healthcare services, e.g., health monitoring, disease diagnosis, and remote surgery [307]. The EECC framework can provide powerful data storage and fast service responses for smart medical devices, as presented in Fig. 19. However, IoT devices in the healthcare ecosystem are usually highly heterogeneous due to different types of medical tasks. An efficient orchestration for different types of healthcare IoT devices is crucial to ensure the quality of service. Particularly, smart healthcare mainly focuses on health monitoring and AI-assisted medical diagnosis.

1) Health Monitoring: Smart health monitoring systems collect health conditions (e.g., heart rate, blood pressure, and sleep status) of patients by employing various wearable devices, sensors, cameras, and other IoT devices, especially for in-home health monitoring for the aging population [308]. A key issue is the data privacy concern. The EECC-based FL framework is characterized by high privacy-preserving and low latency, which is an ideal solution for the health monitoring system. In [284], FedHome learned a shared global model in the cloud from multiple homes at the network edges and achieves data privacy protection by keeping user data locally. By using edge-cloud hybridization mode, existing IoT ecosystems can support real-time biosensor streaming service provisioning and analytics in a resource-constrained environment [285].

2) AI-Assisted Medical Diagnosis: Recently, visual-based healthcare approaches have been explored in disease analysis systems (e.g., electroencephalography [286], CT images [287], and gastroscope [289]), assisting doctors to diagnose diseases more accurately and reduce their workload [288]. Most computer-aided medical systems provide intelligent analysis services that rely on public cloud platforms, suffering from high communication and computing costs. The geographical dispersion and dynamicity of physical nodes also challenge the development of the healthcare system. A promising solution is to employ an intelligent EECC framework for healthcare systems, which is expected to improve the end-to-end performance of next-generation smart healthcare [288].

In the EECC empowered smart healthcare, IoT devices and medical machines collect data on the end side. Then, the basic data analysis and processing are performed at the edge side. Lastly, the cloud server is responsible for system orchestration and management. Based on this collaborative framework, a smart electronic gastroscope system was proposed to realize lesion location and fine-grained disease classification of gastroscopic videos in an online mode [289]. The DL-based CT image analysis were studied in remote diagnostic systems, improving the accuracy of disease classification and shortening the response time [287]. Besides, Wu et al. [290] considered that traditional cloud-based emotion recognition models lack the abilities in characterizing individual variations of the users, and designed a private transfer learning-based emotion recognition structure under the cloud-edge-client collaborations.

E. Real-Time Video Analytics

Real-time video analytics plays important roles in public safety surveillance, autonomous driving, virtual reality (VR) and augmented reality (AR), etc. Generally, accurate video analytics in real-time requires complex image processing algorithms, large memory, and high-performance hardware resources [291]. However, analyzing large-scale live video streams in the cloud is impractical due to high bandwidth consumption, unexpected latency, and reliability issues. On the other hand, edge devices are often incapable of executing complex vision algorithms due to restricted resources. Given the infeasibility of both cloud-only and edge-only solutions, a collaborative end-edge-cloud architecture for large-scale real-time video analytics is necessary, as shown in Fig. 20.

In the EECC architecture, each level is responsible for different functions. Specifically, at the end level, devices such as surveillance cameras are responsible for video capture and

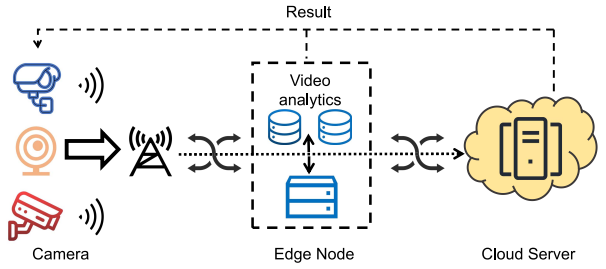


Fig. 20. An illustration of real video analytics.

video compression, etc. At the edge level, the edge layer enables model inference and model selection, and then the requests are forwarded to the backend cloud via a gateway. When an end node is unable to provide accurate analysis, edge nodes can cooperate to provide better services [292], instead of directly delegating to the cloud resources. The cloud is mainly responsible for model training and parameter updating, data storage, and processing complex tasks in collaboration with edge nodes. In the following, we summarize two common application scenarios of real-time video analytics: salient object detection and video query.

1) Salient Object Detection: Salient object detection is a crucial function to estimate human behaviors in various surveillance applications. The authors [291] leveraged collaborative cloud-edge deep reinforcement learning for real-time surveillance video salient object detection. Firstly, the surveillance camera extracts key frames for compressing video data at the edge. Then, the Asynchronous Advantage Actor Critic algorithm schedules object detection tasks adaptively in the edge or cloud. However, real-time object detection in autonomous driving cannot always be guaranteed due to dynamic channel quality. To mitigate this problem, Kim et al. [293] focused on extracting the region of interest (RoI), compressing the image data based on the RoIs, and transmitting the compressed images to the edge or cloud. However, the traditional cloud-edge distributed schemes are unsuitable for cross-camera object detection tasks because of the unbalance of semantic knowledge caused by model training in different scene-specific servers. To tackle this challenge, Gao et al. [294] proposed to build a semantic similarity-based intelligent network, which can make saliency detection in the edge server suitable for its scene, to achieve a balance between global contextual information and local detailed information.

2) Video Query: Another application of video analytics is the real-time query of objects (e.g., cars and people). For example, users input the query command including query objects and locations, the system outputs the video frames with query objects from the recorded video databases. The challenge of video query is to make fast spatio-temporal matching from massive videos. This motivates us to synergize the virtually unlimited cloud resources with agile edge processing to deliver an online real-time video query. In [295], Wang et al. designed an intelligent task allocator to balance the load among different computing nodes with a latency-accuracy tradeoff for real-time queries. Besides, considering the complex interaction and dependency within and across video query pipelines, Zhang et al. [296] designed CEVAS, which can support fine-grained and adaptive partitioning of cloud-edge workloads for multiple concurrent queries. It can reduces costs when compared to a pure cloud scheme and improves the query throughput when compared to an edge-only scheme.

F. Summary and Lessons Learned

In this section, we have discussed five applications empowered by DAI-EECC. Some representative and state-of-the-art references are provided in Table IX. Two key lessons learned are summarized as follows.

1) Customized Distributed AI Algorithms: For all applications we have discussed in this section, the three most important tasks are risk management, privacy preserving, and response latency reduction, which can make the systems smarter, securer, and faster with the assistance of DAI-EECC. However, the unique characteristics of each application motivate us to design customized distributed AI algorithms to satisfy the application requirements. We summarize and compare the design requirements of the distributed AI algorithms for different applications in Table X.

2) Deep Analysis of Practical Deployment: Although many existing works adopt EECC architecture for AI applications, there is currently no experience work to deeply analyze the long-term deployment performance in practical testbeds. The practical deployment performance analysis will provide important guidance for future applications on choosing appropriate distributed learning architectures and learning models, and also help to identify performance bottlenecks for further optimization.

VI. Research Challenges and Open Issues

As discussed in the above sections, DAI-EECC architecture demonstrates its increasingly significant role in promoting intelligent services and applications. Although existing studies provide comprehensive investigations on DAI-EECC from various aspects, the field of study is still in its infancy and needs continuous research efforts to fuel the explosion of distributed AI applications. In this section, we briefly outline some future research directions to foster further research.

A. Service Orchestration and Resource Management in Computing Power Network

The development of EECC makes computing power present a distributed and universal trend. To fully release the potential of distributed computing capabilities, the computing power network is a promising technology [309]. Particularly, the computing power network is a novel infrastructure that flexibly allocates and schedules computing, storage, and communication resources between cloud, edge, and end devices according to service requirements. Currently, the computing power network is in its fancy, many open issues can be studied to enable more efficient service orchestration and resource management. For example, (i) by flexible cross-layer scheduling and orchestration of computing power, how to provide on-demand computing services without paying attention to the location of resources. (ii) How to unify computing power

TABLE IX
THE APPLICATIONS DRIVEN BY END-EDGE-CLOUD COMPUTING INTELLIGENCE

Application scenarios	Ref.	Specific function	End task	Edge task	Cloud task	Evaluation testbed/data
Smart Industry and Manufacturing	[265]	Control	Data collection	Edge data processing, Transmission	Cloud processing	Digital twin 3D printers
	[266]	Monitoring	Local inference	Edge data processing	Architecture search	Industrial sensor monitoring equipment
	[267]	Monitoring	Data sensing	Data storage, compute	Devices management	Real-time intrusion detection system
	[268]	Scheduling	User request	Edge scheduling	Model learning, cloud scheduling	CloudSim simulation platform
	[269]	Prediction	Data generating	Edge forecasting	Cloud forecasting	3 months electricity load data
	[270]	Prediction	Data collection	Data filtering	Data processing, system supervision	Underwater sensors data
	[271]	Defect detection	Data acquisition	Coarse recognition, direction estimation	Online detection	/
	[272]	Fault detection	Data acquisition	Online testing	Offline training	PRONOSTIA data set
ITS	[273]	Autonomous driving	Data sensing	Computing, resource migration	Storage, caching	Sandbox-9 (SB9) on the ORBIT
	[274]	Autonomous driving	Data collection	Behavior evaluation	Model training	Vehicle-edge-cloud architecture through in the campus
	[275]	Traffic flow detection	Data collection	Detection, tracking	Model training	UA-DETRAC data set
	[276]	Video surveillance	Data collection	Data analysis, detection	Model training	/
	[277]	Ridesharing scheduling	Data collection	Request transmission	Vehicle scheduling	/
	[278]	Resource orchestration	Data collection	Request transmission	Policy learning	/
Smart Grid	[279]	Connectivity verification	Data collection	Assistance computation	Identification, recommendation, verification	1 month voltage time-series data
	[280]	Equipment monitoring	Data sensing	Data storage, compute	Devices management	/
	[281]	EV charging scheduling	Data collection	Load prediction	Power supply planning	Distribution network in China
	[282]	EV charging scheduling	Signals receiving	Data aggregation	Scheduling, management	Open source IoT platform, NodeMCU ESP8266-12E
	[283]	EV charging scheduling	Signals receiving	Data aggregation	Scheduling, management	/
Smart Health	[284]	Health monitoring	Data collection	Edge monitoring	Global training, aggregation	MobiAct data set
	[285]	Service provisioning	Data collection	Edge processing	Cloud provisioning	Pulse-sensor data
	[286]	EEG image classification	Data collection	Edge aggregation	Global training, model aggregation	MindBigData EEG data set
	[287]	Remote diagnostic	Data collection	Data mining, local notification	Data analysis, storage, remote interface	SARS-CoV-2 CT-scan data set
	[288]	Assisted healthcare	Measurement	Computing, caching, communication	System management	/
	[289]	Electronic gastroscope disease detection	Data collection	Video management	Disease screening	Real gastroscopic images
	[290]	Emotion recognition	Signals collection	Customize local model	Monitoring, result feedback	SEED and DEAP
Real-time Video Analytics	[291]	Salient object detection	Data collection	Video compression	Object detection	YouTube live video, Kaggle open road surveillance video, SOD
	[292]	Video analysis	Data collection	Preprocessing, model selection, edge inference	DNN computation, model selection, request forwarding	/
	[293]	Object detection	Data collection	Video compression	Detection	KITTI data set
	[294]	Salient object detection	Data collection	Object detection	Model training	DUT-TR, DUT-OMRON, MSRA10K, ECSSD, HKU-IS, PASCAL-S
	[295]	Video query	Query collection	Object detection, task allocator, video classifier	Judicious classification	YouTube live video
	[296]	Online video analytics	Query collection	Edge execution, video partition	Object storage, cloud execution	AWS Lambda

measurement rules by building an abstract model of heterogeneous computing resources. (iii) How to provide unified heterogeneous hardware development tools to reduce the cost of cross-architecture programming for users. Some potential solutions can be considered, such as computing power network node task classification, multi-level computing power trading allocation mechanism [310], elastic capacity expansion and contraction [311].

B. Adaptive Distributed Learning in DAI-EECC

Heterogeneous devices face the problem of insufficient resources in EECC-powered distributed learning. Thus, we should fully consider the computing capability differences of participating devices in distributed training. However, the adaptive distributed training and inference mechanism for heterogeneous EECC has not been thoroughly studied. Therefore, i) how to design customized model structures for different training and inference devices according to the resources and computing capabilities of heterogeneous devices is a key challenge. ii) Since the quality of distributed data and the importance of model parameters are different, how to design efficient aggregation methods for heterogeneous models, as well as optimizing aggregation efficiency and accuracy is worthy to be further explored. iii) Most of the existing studies assume that the model is static during training and

TABLE X
APPLICATION CHARACTERISTICS COMPARISON

Application	Task type	Requirement		
		Model size (Large:L Medium:M Small:S)	Inference latency (High:H Medium:M Low:L)	Security level (High:H Medium:M Low:L)
Smart Industry and Manufacture	Control	L	H	H
	Monitoring	S	L	H
	Detection	M	M	M
	Prediction	M	L	L
ITS	Autonomous driving	L	H	H
	PV scheduling	M	M	M
Smart Grid	Grid operation	M	L	M
Smart Healthcare	Health monitoring	S	M	M
	Medical diagnosis	L	M	M
Video Analytics	SOD	L	H	M
	Video query	M	H	M

cannot adapt according to the capability of equipment. It is another challenge to design the fine-tuning scheme for local models, so that it can adapt to the optimal configuration to realize the performance optimization of the global model. There are some potential solutions: i) adopting model pruning strategy to customize model structures for different training devices; ii) designing an efficient aggregation method for heterogeneous model structures by considering the quality of distributed data and the importance of model parameters; and iii) designing dynamic local model tuning method to optimize the performance of the global model [312]. In addition, another promising future research direction is client scheduling in FL since distributed learning poses different objectives to pure wireless communications when base stations schedule clients [313].

C. End-Edge-Cloud Intelligence in Metaverse

Metaverse is envisioned to be the next-generation Internet after the Web and the mobile network revolutions, in which humans (acting as digital avatars) can interact with other people and software applications in a three-dimensional virtual world by head-mounted display [314]. These services require ultra-low latency and ultra-high bandwidth (e.g., super-realism and zero-latency virtual environments at scale), which presents a new challenge for distributed intelligence. To meet diversified and stringent requirements for different processes in the Metaverse, an efficient orchestrator is a necessity for the interaction between the cloud and edge [315]. Therefore, how to enable the DAI-EECC coordinator to integrate the cloud and edge resources and data processing capabilities for users' seamless experience in Metaverse is a promising direction.

D. Advanced Security and Privacy-Preserving Technologies

Although there are many existing works solving security and privacy threats in the model training and inference process. We still face several challenges. (i) In the training phase, DP and encryption technologies protect the system with sacrifices of efficiency or precision, which are inefficient for distributed learning with limited resources or small-scale data sets. Moreover, the high computation cost of existing defense methods leads to difficulties for model deployment on resource-limited end devices. Therefore, a more lightweight privacy or encryption algorithm that can support nonlinear operation is essential for the model training in the DAI-EECC architecture [316]. (ii) In the inference phase, a well-trained model will be released to the terminal devices for inference tasks, anyone can use the model without permission, which violates the intellectual property rights of the model owner. As a result, model authorization is a future direction needed to be studied. (iii) Besides, the trusted execution environment (TEE) is a useful technology that provides users with confidentiality and integrity (e.g., face recognition). However, TEEs have the disadvantages of low extensibility and poor performance. It is also worth investigating how to improve the performance of AI model training and inference in TEE [317].

E. Cooperative Computing of Coprocessors

To meet the computing requirements of AI technologies for hardware resources, accelerators for AI models have been developed rapidly, from CPU to GPU, TPU, DPU (Deep learning Processing Unit), and NPU. However, due to the limitation that the AI compiler cannot support the cooperative work of multiple coprocessors, these accelerators are independent of each other in the process of model training and inference, resulting in a waste of computing resources. Therefore, it would be an attractive direction to design AI compilers that can compatibly and concurrently process these coprocessors to support AI training and inference tasks [318].

VII. CONCLUSION

Driven by the flourishing of both AI and distributed computing paradigms, there is an urgent demand to integrate distributed AI and hierarchical computing paradigms to promote the rapid development of AI applications, especially in delay-sensitive scenarios. In this survey, we have concentrated on the DAI-EECC architecture to comprehensively introduce and discuss fundamental technologies in supporting distributed AI, and the state-of-the-art optimization technologies, as well as the security and privacy protection technologies to address the major challenges in DAI-EECC. Additionally, we have reviewed several emerging applications and highlighted the remaining research challenges and open issues in DAI-EECC. We hope this survey will provide a clear overview for readers to grasp the current status of the emerging field of study, and inspire continuous research efforts on DAI-EECC to promote future distributed AI applications and services.

REFERENCES

[1] L. S. Vailshery. "IoT and Non-IoT Connections Worldwide 2010–2025." Mar. 2021. [Online]. Available: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/
[2] R. Badnakhe. "IoT Is Not a Buzzword But Necessity." Feb. 2021. [Online]. Available: https://www.iotcentral.io/blog/iot-is-not-a-buzzword-but-necessity

[3] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L.-C. Wang, "Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 44–52, Jun. 2019.

[4] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Netw.*, vol. 34, no. 6, pp. 272–280, Nov./Dec. 2020.

[5] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13849–13875, Sep. 2021.

[6] Y. Wu, "Cloud-edge orchestration for the Internet of Things: Architecture and AI-powered data processing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12792–12805, Aug. 2021.

[7] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surveys*, vol. 52, no. 6, pp. 1–36, 2019.

[8] J. Yao et al., "Edge-cloud polarization and collaboration: A comprehensive survey for AI," *IEEE Trans. Knowl. Data Eng.*, early access, May 27, 2022, doi: 10.1109/TKDE.2022.3178211.

[9] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[10] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A review on computational intelligence techniques in cloud and edge computing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 6, pp. 742–763, Dec. 2020.

[11] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li, "A survey on edge and edge-cloud computing assisted cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7806–7819, Nov. 2021.

[12] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[13] 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.

[14] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[15] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing AI to edge: From deep learning's perspective," *Neurocomputing*, vol. 485, pp. 297–320, May 2022.

[16] D. C. Nguyen et al., "Enabling AI in future wireless networks: A data life cycle perspective," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 553–595, 1st Quart., 2020.

[17] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surveys*, vol. 54, no. 8, pp. 1–37, 2021.

[18] C. Ma et al., "Trusted AI in multi-agent systems: An overview of privacy and security for distributed learning," 2022, *arXiv:2202.09027*.

[19] N. L. da Fonseca and R. Boutaba, *Cloud Services, Networking, and Management*. Hoboken, NJ, USA: Wiley, 2015.

[20] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[21] S. Yue et al., "TODG: Distributed task offloading with delay guarantees for edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1650–1665, Jul. 2022.

[22] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020.

[23] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, "Efficient dependent task offloading for multiple applications in MEC-cloud system," *IEEE Trans. Mobile Comput.*, early access, Oct. 11, 2022, doi: 10.1109/TMC.2021.3119200.

[24] J. Zhang and D. Tao, "Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7789–7817, May 2021.

[25] 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, Jun. 2017.

[26] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," 2016, *arXiv:1604.00981*.

[27] K. S. Chahal, M. S. Grover, K. Dey, and R. R. Shah, "A Hitchhiker's guide on distributed training of deep neural networks," *J. Parallel Distrib. Comput.*, vol. 137, pp. 65–76, Mar. 2020.

[28] N. Ferdinand, B. Gharachorloo, and S. C. Draper, "Anytime exploitation of stragglers in synchronous stochastic gradient descent," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2017, pp. 141–146.

[29] W. Zhang, S. Gupta, X. Lian, and J. Liu, "Staleness-aware async-SGD for distributed deep learning," 2015, *arXiv:1511.05950*.

[30] S. Zheng et al., "Asynchronous stochastic gradient descent with delay compensation," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 4120–4129.

[31] T. Sun, R. Hannah, and W. Yin, "Asynchronous coordinate descent under more realistic assumptions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6182–6190.

[32] Y. Chen, K. Yuan, Y. Zhang, P. Pan, Y. Xu, and W. Yin, "Accelerating Gossip SGD with periodic global averaging," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 1791–1802.

[33] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized Gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.

[34] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," 2016, *arXiv:1611.09726*.

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

[36] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and Gossip algorithms with compressed communication," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 3478–3487.

[37] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 344–353.

[38] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[39] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.

[40] Y. Ma, D. Yu, T. Wu, and H. Wang, "PaddlePaddle: An open-source deep learning platform from industrial practice," *Front. Data Domput.*, vol. 1, no. 1, pp. 105–115, 2019.

[41] Huawei. "MindSpore." 2020. [Online]. Available: https://github.com/mindspore-ai/mindspore

[42] MEGVII. "MegEngine: A Fast, Scalable and Easy-to-Use Deep Learning Framework." 2020. [Online]. Available: https://github.com/MegEngine/MegEngine

[43] A. Sergeev and M. D. Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," 2018, *arXiv:1802.05799*.

[44] F. Seide and A. Agarwal, "CNTK: Microsoft's open-source deep-learning toolkit," in *Proc. 22nd ACM Int. Conf. Knowl. Disc. Data Min. (SIGKDD)*, 2016, p. 2135.

[45] Xiaomi. "MACE." 2018. [Online]. Available: https://github.com/XiaoMi/mace

[46] "Deeplearning4j." Aug. 2019. [Online]. Available: https://news.microsoft.com/features/democratizing-ai/

[47] Google. "JAX: Autograd and XLA." 2022. [Online]. Available: https://github.com/google/jax

[48] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters," in *Proc. 14th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2020, pp. 463–479.

[49] A. Markham and Y. Jia, *Caffe2: Portable High-Performance Deep Learning Framework From Facebook*, NVIDIA, Santa Clara, CA, USA, 2017.

[50] F. Chollet. "Keras." Aug. 2021. [Online]. Available: https://keras.io/

[51] M. Innes, "FLUX: Elegant machine learning with julia," *J. Open Source Softw.*, vol. 3, no. 25, p. 602, 2018.

[52] Intel. "Intel AI. PlaidML." Aug. 2018. [Online]. Available: https://www.intel.ai/reintroducing-plaidmlL

[53] D. Liu, X. Chen, Z. Zhou, and Q. Ling, "HierTrain: Fast hierarchical edge AI learning with hybrid parallelism in mobile-edge-cloud computing," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 634–645, 2020.

[54] AWS. "Introduction to Model Parallelism." Aug. 2021. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/model-parallel-intro.html

[55] J. Dean et al., "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 25, 2012, pp. 1223–1231.

[56] S. Sun, W. Chen, J. Bian, X. Liu, and T.-Y. Liu, "Slim-DP: A multi-agent system for communication-efficient distributed deep learning," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 721–729.

[57] Y. Kang et al., "NeuroSurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, 2017.

[58] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2017, *arXiv:1712.01887*.

[59] J. Fang, H. Fu, G. Yang, and C.-J. Hsieh, "RedSync: Reducing synchronization bandwidth for distributed deep learning training system," *J. Parallel Distrib. Comput.*, vol. 133, pp. 30–39, Nov. 2019.

[60] W. Wen et al., "TernGrad: Ternary gradients to reduce communication in distributed deep learning," 2017, *arXiv:1705.07878*.

[61] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1709–1720.

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

[63] S. B. Akintoye, L. Han, X. Zhang, H. Chen, and D. Zhang, "A hybrid parallelization approach for distributed and scalable deep learning," 2021, *arXiv:2104.05035*.

[64] S. Pal et al., "Optimizing multi-GPU parallelization strategies for deep learning training," *IEEE Micro*, vol. 39, no. 5, pp. 91–101, Sep./Oct. 2019.

[65] J. Geng, D. Li, and S. Wang, "Horizontal or vertical? A hybrid approach to large-scale distributed machine learning," in *Proc. 10th Workshop Sci. Cloud Comput.*, 2019, pp. 1–4.

[66] S. Yao et al., "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proc. 18th Conf. Embedded Netw. Sensor Syst.*, 2020, pp. 476–488.

[67] M. Li, Y. Li, Y. Tian, L. Jiang, and Q. Xu, "AppealNet: An efficient and highly-accurate edge/cloud collaborative architecture for DNN inference," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 409–414.

[68] M. Xue, H. Wu, G. Peng, and K. Wolter, "DDPQN: An efficient DNN offloading strategy in local-edge-cloud collaborative environments," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 640–655, Mar./Apr. 2022.

[69] M. Xue, H. Wu, R. Li, M. Xu, and P. Jiao, "EosDNN: An efficient offloading scheme for DNN inference acceleration in local-edge-cloud collaborative environments," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 248–264, Mar. 2022.

[70] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Comput. Surveys*, to be published.

[71] J. Kim, Y. Park, G. Kim, and S. J. Hwang, "SplitNet: Learning to semantically split deep networks for parameter reduction and model parallelization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1866–1874.

[72] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.

[73] R. A. Cohen, H. Choi, and I. V. Bajić, "Lightweight compression of neural network feature tensors for collaborative intelligence," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, 2020, pp. 1–6.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[75] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 2818–2826.

[76] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeuraIPS)*, vol. 28, 2015, pp. 91–99.

[77] A. E. Eshratifar, A. Esmaili, and M. Pedram, "BottleNet: A deep learning architecture for intelligent mobile cloud computing services," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, 2019, pp. 1–6.

[78] Y. Matsubara, S. Baidya, D. Callegaro, M. Levorato, and S. Singh, "Distilled split deep neural networks for edge-assisted real-time systems," in *Proc. Workshop Hot Topics Video Anal. Intell. Edges*, 2019, pp. 21–26.

[79] Y. Matsubara and M. Levorato, "Neural compression and filtering for edge-assisted real-time object detection in challenged networks," in *Proc. IEEE 25th Int. Conf. Pattern Recognit. (ICPR)*, 2021, pp. 2272–2279.

[80] S. Li, Q. Qi, J. Wang, H. Sun, Y. Li, and F. R. Yu, "GGS: General gradient sparsification for federated learning in edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–7.

[81] S. Shi et al., "Communication-efficient distributed deep learning with merged gradient sparsification on GPUs," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2020, pp. 406–415.

[82] L. Abrahamyan, Y. Chen, G. Bekoulis, and N. Deligiannis, "Learned gradient compression for distributed deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 10, 2021, doi: 10.1109/TNNLS.2021.3084806.

[83] C.-Y. Chen et al., "ScaleCom: Scalable sparsified gradient compression for communication-efficient distributed training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 13551–13563.

[84] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2019, pp. 1–8.

[85] Y. He, M. Zenk, and M. Fritz, "CosSGD: Nonlinear quantization for communication-efficient federated learning," 2020, *arXiv:2012.08241*.

[86] A. Abdi and F. Fekri, "Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3105–3112.

[87] A. Abdi, "Indirect stochastic gradient quantization and its application in distributed deep learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3113–3120.

[88] J. Guo et al., "Accelerating distributed deep learning by adaptive gradient quantization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2020, pp. 1603–1607.

[89] A. Xu, Z. Huo, and H. Huang, "Optimal gradient quantization condition for communication-efficient distributed training," 2020, *arXiv:2002.11082*.

[90] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 14668–14679.

[91] X. Ma et al., "Pconv: The missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 5117–5124.

[92] W. Niu et al., "GRIM: A general, real-time deep learning inference framework for mobile devices based on fine-grained structured weight sparsity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6224–6239, Oct. 2022.

[93] P. Peng, M. You, W. Xu, and J. Li, "Fully integer-based quantization for mobile convolutional neural network inference," *Neurocomputing*, vol. 432, pp. 194–205, Apr. 2021.

[94] S.-E. Chang et al., "Mix and match: A novel FPGA-centric deep neural network quantization framework," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2021, pp. 208–220.

[95] Q. Sun, L. Jiao, Y. Ren, X. Li, F. Shang, and F. Liu, "Effective and fast: A novel sequential single path search for mixed-precision quantization," 2021, *arXiv:2103.02904*.

[96] W. Tang, X. Wei, and B. Li, "Automated model compression by jointly applied pruning and quantization," 2020, *arXiv:2011.06231*.

[97] J. Wu et al., "PocketFlow: An automated framework for compressing and accelerating deep neural networks," in *Proc. NIPS*, 2018, pp. 1–8.

[98] 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.*, vol. 32, 2018, pp. 2827–2835.

[99] T. Vogels, S. P. Karinireddy, and M. Jaggi, "PowerSGD: Practical low-rank gradient compression for distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019, pp. 14236–14245.

[100] S. Shi et al., "A distributed synchronous SGD algorithm with global top-k sparsification for low bandwidth networks," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2019, pp. 2238–2247.

[101] A. M Abdelmoniem, A. Elzanaty, M.-S. Alouini, and M. Canini, "An efficient statistical-based gradient compression technique for distributed training systems," in *Proc. Mach. Learn. Syst.*, vol. 3, 2021, pp. 1–9.

[102] J.-H. Luo and J. Wu, "An entropy-based pruning method for CNN compression," 2017, *arXiv:1706.05791*.

[103] M. M. Pasandi, M. Hajabdollahi, N. Karimi, and S. Samavi, "Modeling of pruning techniques for deep neural networks simplification," 2020, *arXiv:2001.04062*.

[104] W. Niu, P. Zhao, Z. Zhan, X. Lin, Y. Wang, and B. Ren, "Towards real-time DNN inference on mobile platforms with model pruning and compiler optimization," 2020, *arXiv:2004.11250*.
[105] 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, *arXiv:1602.02830*.
[106] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2016, *arXiv:1612.01064*.
[107] Y. H. Oh et al., "A portable, automatic data qantizer for deep neural networks," in *Proc. 27th Int. Conf. Parallel Archit. Compilation Techn.*, 2018, pp. 1–14.
[108] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2015, *arXiv:1510.00149*.
[109] Q. He, M. Dong, and L. Schwiebert, "Octave deep compression: In-parallel pruning-quantization on different frequencies," in *Proc. IEEE 22nd Int. Conf. Inf. Reuse Integr. Data Sci. (IRI)*, 2021, pp. 184–192.
[110] P.-H. Yu, S.-S. Wu, J. P. Klopp, L.-G. Chen, and S.-Y. Chien, "Joint pruning & Quantization for extremely sparse neural networks," 2020, *arXiv:2010.01892*.
[111] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: Automl for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 784–800.
[112] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[113] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.
[114] A. Howard et al., "Searching for MobileNetv3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1314–1324.
[115] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5 MB model size," 2016, *arXiv:1602.07360*.
[116] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 6848–6856.
[117] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1251–1258.
[118] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More features from cheap operations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1580–1589.
[119] C. Cui et al., "PP-LCNet: A lightweight CPU convolutional neural network," 2021, *arXiv:2109.15099*.
[120] J. Zhu, X. Lou, and W. Ye, "Lightweight deep learning model in mobile-edge computing for radar-based human activity recognition," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12350–12359, Aug. 2021.
[121] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.
[122] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.
[123] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.
[124] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2019.
[125] N. Mhaisen, A. A. Abdellatif, A. Mohamed, A. Erbad, and M. Guizani, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 55–66, Jan./Feb. 2022.
[126] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Hierarchical quantized federated learning: Convergence analysis and system design," 2021, *arXiv:2103.14272*.
[127] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Local averaging helps: Hierarchical federated learning and convergence analysis," 2020, *arXiv:2010.12998*.
[128] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2021, pp. 1–10.
[129] Y. Deng et al., "SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2021, pp. 24–34.
[130] W. Luping, W. Wei, and L. Bo, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2019, pp. 954–964.
[131] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.
[132] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.
[133] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Depend. Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2019.
[134] K. Bonawitz et al., "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 374–388.
[135] S. Duan, D. Zhang, Y. Wang, L. Li, and Y. Zhang, "JointRec: A deep-learning-based joint cloud video recommendation framework for mobile IoT," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1655–1666, Mar. 2020.
[136] Y. Wu, Y. Song, T. Wang, L. Qian, and T. Q. Quek, "Non-orthogonal multiple access assisted federated learning via wireless power transfer: A cost-efficient approach," *IEEE Trans. Commun.*, vol. 70, no. 4, pp. 2853–2869, Apr. 2022.
[137] K. Xie et al., "Efficient federated learning with spike neural networks for traffic sign recognition," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9980–9992, Sep. 2022.
[138] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.
[139] Y. Deng et al., "Auction: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1996–2009, Aug. 2022.
[140] Y. Wu, Y. Song, T. Wang, M. Dai, and T. Q. Quek, "Simultaneous wireless information and power transfer assisted federated learning via non-orthogonal multiple access," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1846–1861, Sep. 2022.
[141] J. Kang et al., "Communication-efficient and cross-chain empowered federated learning for artificial intelligence of things," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 2966–2977, Sep./Oct. 2022.
[142] Q. Li et al., "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 2, 2021, doi: 10.1109/TKDE.2021.3124599.
[143] S. Yue, J. Ren, J. Xin, D. Zhang, Y. Zhang, and W. Zhuang, "Efficient federated meta-learning over multi-access wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1556–1570, May 2022.
[144] W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, C. Miao, and D. I. Kim, "Dynamic edge association and resource allocation in self-organizing hierarchical federated learning networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3640–3653, Dec. 2021.
[145] W. Y. B. Lim et al., "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9575–9588, Oct. 2020.
[146] Y. Deng et al., "FAIR: Quality-aware federated learning with precise user incentive and model aggregation," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2021, pp. 1–10.
[147] S. Warnat-Herresthal et al., "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.
[148] Y. Park, D.-J. Han, D.-Y. Kim, J. Seo, and J. Moon, "Few-round learning for federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 28612–28622.
[149] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," 2019, *arXiv:1902.11175*.
[150] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu, "Distilled one-shot federated learning," 2020, *arXiv:2009.07999*.
[151] A. Kasturi, A. R. Ellore, and C. Hota, "Fusion learning: A one shot federated learning," in *Proc. Int. Conf. Comput. Sci.*, 2020, pp. 424–436.
[152] A. Kasturi, A. R. Ellore, P. Saxena, and C. Hota, "Hybrid fusion learning: A hierarchical learning model for distributed systems," in *Proc. 4th Int. Workshop Embedded Mobile Deep Learn.*, 2020, pp. 1–6.
[153] Google. "EdgeTPU." Aug. 2018. [Online]. Available: https://cloud.google.com/edge-tpu/

[154] HiSilicon. "The World's First Full-Stack All-Scenario AI Chip." Aug. 2018. [Online]. Available: http://www.hisilicon.com/en/Products/ProductList/Ascend

[155] S. Jiang et al., "Accelerating mobile applications at the network edge with software-programmable FPGAs," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 55–62.

[156] M. Wang, S. Ding, T. Cao, Y. Liu, and F. Xu, "AsyMo: Scalable and efficient deep-learning inference on asymmetric mobile CPUs," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 215–228.

[157] S. Jiang, L. Ran, T. Cao, Y. Xu, and Y. Liu, "Profiling and optimizing deep learning inference on mobile GPUs," in *Proc. 11th ACM SIGOPS Asia–Pac. Workshop Syst.*, 2020, pp. 75–81.

[158] S. Kim, S. Oh, and Y. Yi, "Minimizing GPU kernel launch overhead in deep learning inference on mobile GPUs," in *Proc. 22nd Int. Workshop Mobile Comput. Syst. Appl.*, 2021, pp. 57–63.

[159] J. Lee, Y. Liu, and Y. Lee, "ParallelFusion: Towards maximum Utilization of mobile GPU for DNN inference," in *Proc. 5th Int. Workshop Embedded Mobile Deep Learn.*, 2021, pp. 25–30.

[160] C. Oh, J. So, S. Kim, and Y. Yi, "Exploiting activation sparsity for fast CNN inference on mobile GPUs," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 5s, pp. 1–25, 2021.

[161] J. Zhang et al., "MobiPose: Real-time multi-person pose estimation on mobile devices," in *Proc. 18th Conf. Embedded Netw. Sensor Syst.*, 2020, pp. 136–149.

[162] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.

[163] E. Nurvitadhi et al., "Can FPGAs beat GPUs in accelerating next-generation deep neural networks?" in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2017, pp. 5–14.

[164] L. Du et al., "A reconfigurable streaming deep convolutional neural network accelerator for Internet of Things," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 198–208, Jan. 2018.

[165] R. Morcel et al., "FeatherNet: An accelerated convolutional neural network design for resource-constrained FPGAs," *ACM Trans. Reconfig. Technol. Syst.*, vol. 12, no. 2, pp. 1–27, 2019.

[166] Y. Yu, T. Zhao, K. Wang, and L. He, "Light-OPU: An FPGA-based overlay processor for lightweight convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2020, pp. 122–132.

[167] N. D. Lane et al., "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, 2016, pp. 1–12.

[168] M. Alzantot, Y. Wang, Z. Ren, and M. B. Srivastava, "RStensorFlow: GPU enabled tensorflow for deep learning on commodity android devices," in *Proc. 1st Int. Workshop Deep Learn. Mobile Syst. Appl.*, 2017, pp. 7–12.

[169] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "DeepCache: Principled cache for mobile deep vision," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 129–144.

[170] K. Lee, J. Kong, and A. Munir, "HW/SW co-design of cost-efficient CNN inference for cognitive IoT," in *Proc. IEEE 4th Int. Conf. Intell. Comput. Data Sci. (ICDS)*, 2020, pp. 1–6.

[171] J. Haris, P. Gibson, J. Cano, N. B. Agostini, and D. Kaeli, "SECDA: Efficient hardware/software co-design of FPGA-based DNN accelerators for edge inference," in *Proc. IEEE 33rd Int. Symp. Comput. Architect. High Perform. Comput. (SBAC-PAD)*, 2021, pp. 33–43.

[172] M. Liu, S. Luo, K. Han, B. Yuan, R. F. DeMara, and Y. Bai, "An efficient real-time object detection framework on resource-constricted hardware devices via software and hardware co-design," in *Proc. IEEE 32nd Int. Conf. Appl. Specific Syst. Archit. Process. (ASAP)*, 2021, pp. 77–84.

[173] X. Luo, D. Liu, S. Huai, and W. Liu, "HSCoNAS: Hardware-software co-design of efficient DNNs via neural architecture search," in *Proc. IEEE Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2021, pp. 418–421.

[174] A. Ahmad, M. A. Pasha, and G. J. Raza, "Accelerating tiny YOLOv3 using FPGA-based hardware/software co-design," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2020, pp. 1–5.

[175] S. Wiedemann et al., "FantastIC4: A hardware-software co-design approach for efficiently running 4bit-compact multilayer perceptrons," *IEEE Open J. Circuits Syst.*, vol. 2, pp. 407–419, 2021.

[176] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proc. ACM Symp. Cloud Comput.*, 2018, pp. 401–411.

[177] Y. Huang, F. Wang, F. Wang, and J. Liu, "DeePar: A hybrid device-edge-cloud execution framework for mobile deep learning applications," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2019, pp. 892–897.

[178] J. Mao et al., "MEDNN: A distributed mobile system with enhanced partition and deployment for large-scale DNNs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2017, pp. 751–756.

[179] 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.

[180] 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.

[181] L. Wang et al., "Context-aware deep model compression for edge cloud computing," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2020, pp. 787–797.

[182] M. Sbai, M. R. U. Saputra, N. Trigoni, and A. Markham, "Cut, distil and encode (CDE): Split cloud-edge deep inference," in *Proc. 18th Annu. IEEE Int. Conf. Sens. Commun. Netw. (SECON)*, 2021, pp. 1–9.

[183] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, and Y. Zhang, "Auto-split: A general framework of collaborative edge-cloud AI," in *Proc. 27th ACM SIGKDD Conf. Knowl. Disc. Data Min.*, 2021, pp. 2543–2553.

[184] S. Zhang et al., "Towards real-time cooperative deep inference over the cloud and edge end devices," *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 4, no. 2, pp. 1–24, 2020.

[185] 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. Workshops (ICC Workshops)*, 2020, pp. 1–6.

[186] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, "Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2783–2797, Dec. 2020.

[187] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Cachier: Edge-caching for recognition applications," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 276–286.

[188] T. Guo, R. J. Walls, and S. S. Ogden, "EdgeServe: Efficient deep learning model caching at the edge," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, 2019, pp. 313–315.

[189] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, 2016, pp. 2464–2469.

[190] S. Laskaridis, A. Kouris, and N. D. Lane, "Adaptive inference through early-exit networks: Design, challenges and directions," in *Proc. 5th Int. Workshop Embedded Mobile Deep Learn.*, 2021, pp. 1–6.

[191] R. G. Pacheco, F. D. Oliveira, and R. S. Couto, "Early-exit deep neural networks for distorted images: Providing an efficient edge offloading," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[192] C. Li, H. Xu, Y. Xu, Z. Wang, and L. Huang, "DNN inference acceleration with partitioning and early exiting in edge computing," in *Proc. Int. Conf. Wireless Algorithms Syst. Appl.*, 2021, pp. 465–478.

[193] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: Synergistic progressive inference of neural networks over device and cloud," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–15.

[194] J. Shao, H. Zhang, Y. Mao, and J. Zhang, "Branchy-GNN: A device-edge co-inference framework for efficient point cloud processing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021, pp. 8488–8492.

[195] W. Ju, W. Bao, D. Yuan, L. Ge, and B. B. Zhou, "Learning early exit for deep neural network inference on mobile devices through multi-armed bandits," in *Proc. IEEE/ACM 21st Int. Symp. Cluster Cloud Internet Comput. (CCGrid)*, 2021, pp. 11–20.

[196] Z. Wang, W. Bao, D. Yuan, L. Ge, N. H. Tran, and A. Y. Zomaya, "SEE: Scheduling early exit for mobile DNN inference during service outage," in *Proc. 22nd Int. ACM Conf. Model. Anal. Simulat. Wireless Mobile Syst.*, 2019, pp. 279–288.

[197] L. Zhang, L. Chen, and J. Xu, "Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning," in *Proc. Web Conf.*, 2021, pp. 3111–3123.

[198] R. G. Pacheco, K. Bochie, M. S. Gilbert, R. S. Couto, and M. E. M. Campista, "Towards edge computing using early-exit convolutional neural networks," *Information*, vol. 12, no. 10, p. 431, 2021.

[199] P. Subedi, J. Hao, I. K. Kim, and L. Ramaswamy, "AI multi-tenancy on edge: Concurrent deep learning model executions and dynamic model placements on edge devices," in *Proc. IEEE 14th Int. Conf. Cloud Comput. (CLOUD)*, 2021, pp. 31–42.

[200] B. Fang, X. Zeng, and M. Zhang, "NestDNN: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 115–127.

[201] D. Narayanan, K. Santhanam, A. Phanishayee, and M. Zaharia, "Accelerating deep learning workloads through efficient multi-model execution," in *Proc. NeurIPS Workshop Syst. Mach. Learn.*, 2018, p. 20.

[202] A. H. Jiang et al., "MainStream: Dynamic stem-sharing for multi-tenant video processing," in *Proc. USENIX Annu. Tech. Conf.*, 2018, pp. 29–42.

[203] A. Mathur, N. D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, and F. Kawsar, "DeepEye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware," in *Proc. 15th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2017, pp. 68–81.

[204] B. C. Şenel, M. Mouchet, J. Cappos, O. Fourmaux, T. Friedman, and R. Mcgeer, "EdgeNet: A multi-tenant and multi-provider edge cloud," in *Proc. 4th Int. Workshop Edge Syst. Anal. Netw.*, 2021, pp. 49–54.

[205] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Res. Comput. Security*, 2020, pp. 480–501.

[206] B. Han, I. W. Tsang, and L. Chen, "On the convergence of a family of robust losses for stochastic gradient descent," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2016, pp. 665–680.

[207] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th USENIX Security Symp. (USENIX Security)*, 2020, pp. 1605–1622.

[208] M. T. Hossain, S. Islam, S. Badsha, and H. Shen, "DESMP: Differential privacy-exploited stealthy model poisoning attacks in federated learning," in *Proc. 17th Int. Conf. Mobility Sens. Netw. (MSN)*, 2021, pp. 167–174.

[209] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 118–128.

[210] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[211] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3521–3530.

[212] S. Awan, B. Luo, and F. Li, "CONTRA: Defending against poisoning attacks in federated learning," in *Proc. Eur. Symp. Res. Comput. Security*, 2021, pp. 455–475.

[213] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," 2019, *arXiv:1909.05125*.

[214] B. Wang et al., "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Security Privacy (SP)*, 2019, pp. 707–723.

[215] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" 2019, *arXiv:1911.07963*.

[216] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2020, pp. 2938–2948.

[217] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. Int. Symp. Res. Attacks Intrusions Defenses*, 2018, pp. 273–294.

[218] M. Kuzlu, C. Fair, and O. Guler, "Role of artificial intelligence in the Internet of Things (IoT) cybersecurity," *Disc. Internet Things*, vol. 1, no. 1, pp. 1–14, 2021.

[219] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Analysis of DDoS-capable IoT malwares," in *Proc. IEEE Feder. Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, 2017, pp. 807–816.

[220] Z. Cekerevac, Z. Dvorak, L. Prigoda, and P. Cekerevac, "Internet of Things and the man-in-the-middle attacks—Security and economic risks," *MEST J.*, vol. 5, no. 2, pp. 15–25, 2017.

[221] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in IoT networks," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2019, pp. 452–457.

[222] J. Pan and Z. Yang, "Cybersecurity challenges and opportunities in the new' edge computing + IoT' world," in *Proc. ACM Int. Workshop Security Softw. Defined Netw. Netw. Function Virtualization*, 2018, pp. 29–32.

[223] Y. Ding and H. Sato, "Derepo: A distributed privacy-preserving data repository with decentralized access control for smart health," in *Proc. 7th IEEE Int. Conf. Cyber Security Cloud Comput. (CSCloud) Proc. 6th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, 2020, pp. 29–35.

[224] C. Pahl, N. E. Ioini, S. Helmer, and B. Lee, "An architecture pattern for trusted orchestration in IoT edge clouds," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2018, pp. 63–70.

[225] H. Guo, W. Li, M. Nejad, and C.-C. Shen, "Access control for electronic health records with hybrid blockchain-edge architecture," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, 2019, pp. 44–51.

[226] Y. Ren, F. Zhu, J. Qi, J. Wang, and A. K. Sangaiah, "Identity management and access control based on blockchain under edge computing for the industrial Internet of Things," *Appl. Sci.*, vol. 9, no. 10, p. 2058, 2019.

[227] J. Zhang, J. Zhang, J. Chen, and S. Yu, "GAN enhanced membership inference: A passive local attack in federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[228] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[229] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017, *arXiv:1710.06963*.

[230] D. Wang, J. Ren, Z. Wang, X. Pang, Y. Zhang, and X. S. Shen, "Privacy-preserving streaming truth discovery in crowdsourcing with differential privacy," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3757–3772, Oct. 2022.

[231] Y. Zhao et al., "Local differential privacy-based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, Jun. 2021.

[232] L. Sun, J. Qian, X. Chen, and P. S. Yu, "LDP-FL: Practical private aggregation in federated learning with local differential privacy," 2020, *arXiv:2007.15789*.

[233] A. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. T. Suresh, "Shuffled model of differential privacy in federated learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2021, pp. 2521–2529.

[234] T. Wang et al., "Improving utility and security of the shuffler-based differential privacy," 2019, *arXiv:1908.11515*.

[235] Z. Jiang, W. Wang, and Y. Liu, "FLASHE: Additively symmetric homomorphic encryption for cross-silo federated learning," 2021, *arXiv:2109.00675*.

[236] Y. Aono et al., "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[237] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6532–6542, Oct. 2020.

[238] B. Jia, X. Zhang, J. Liu, Y. Zhang, K. Huang, and Y. Liang, "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4049–4058, Jun. 2022.

[239] T. Li, Z. Liu, V. Sekar, and V. Smith, "Privacy for free: Communication-efficient learning with differential privacy using sketches," 2019, *arXiv:1911.00972*.

[240] D. Wang, J. Ren, Z. Wang, Y. Zhang, and X. S. Shen, "PrivStream: A privacy-preserving inference framework on IoT streaming data at the edge," *Inf. Fusion*, vol. 80, pp. 282–294, Apr. 2022.

[241] H. Li and T. Han, "An end-to-end encrypted neural network for gradient updates transmission in federated learning," 2019, *arXiv:1908.08340*.

[242] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Security Privacy (SP)*, 2019, pp. 739–753.

[243] L. Lyu and C. Chen, "A novel attribute reconstruction attack in federated learning," 2021, *arXiv:2108.06910*.

[244] T. Orekondy, S. J. Oh, Y. Zhang, B. Schiele, and M. Fritz, "Gradient-leaks: Understanding and controlling deanonymization in federated learning," 2018, *arXiv:1805.05838*.

[245] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2512–2520.

[246] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 603–618.

[247] T. Miura, S. Hasegawa, and T. Shibahara, "MEGEX: Data-free model extraction attack against gradient-based explainable AI," 2021, *arXiv:2107.08909*.

[248] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in *Proc. 29th USENIX Security Symp. (USENIX Security)*, 2020, pp. 2003–2020.

[249] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu, "Hermes attack: Steal DNN models with lossless inference accuracy," in *Proc. 30th USENIX Security Symp. (USENIX Security)*, 2021, pp. 1–9.

[250] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11365–11375, Jul. 2022.

[251] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2019, pp. 233–239.

[252] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3310–3322, Mar. 2021.

[253] C. Xie, S. Koyejo, and I. Gupta, "Practical distributed learning: Secure machine learning with communication-efficient local updates," in *Proc. Eur. Conf. Mach. Learn. Principle Pract. Knowl. Disc. Databases (ECML PKDD)*, 2019, p. 19.

[254] C. Ma et al., "When federated learning meets blockchain: A new distributed learning paradigm," *IEEE Comput. Intell. Mag.*, vol. 17, no. 3, pp. 26–33, Aug. 2022.

[255] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Security Privacy (SP)*, 2017, pp. 3–18.

[256] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "CloudLeak: Large-scale deep learning models stealing through adversarial examples," in *Proc. NDSS*, 2020, pp. 1–16.

[257] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1310–1321.

[258] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-fed: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst. Anal. Netw.*, 2020, pp. 61–66.

[259] B. Balle, J. Bell, A. Gascón, and K. Nissim, "The privacy blanket of the shuffle model," in *Proc. Annu. Int. Cryptol. Conf.*, 2019, pp. 638–667.

[260] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. IEEE 27th Annu. Symp. Found. Comput. Sci. (SFCS)*, 1986, pp. 162–167.

[261] Q. Zhang, S. Jing, C. Zhao, B. Zhang, and Z. Chen, "Efficient federated learning framework based on multi-key homomorphic encryption," in *Proc. Int. Conf. P2P Parallel Grid Cloud Internet Comput.*, 2021, pp. 88–105.

[262] Z. L. Jiang, H. Guo, Y. Pan, Y. Liu, X. Wang, and J. Zhang, "Secure neural network in federated learning with model aggregation under multiple keys," in *Proc. 8th IEEE Int. Conf. Cyber Security Cloud Comput. (CSCloud) 7th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, 2021, pp. 47–52.

[263] S. Truex et al., "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Security*, 2019, pp. 1–11.

[264] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Security Privacy (SP)*, 2019, pp. 691–706.

[265] L. Guo, Y. Cheng, Y. Zhang, Y. Liu, C. Wan, and J. Liang, "Development of cloud-edge collaborative digital twin system for FDM additive manufacturing," in *Proc. IEEE 19th Int. Conf. Ind. Inf. (INDIN)*, 2021, pp. 1–6.

[266] H. Lu et al., "An adaptive neural architecture search design for collaborative edge-cloud computing," *IEEE Netw.*, vol. 35, no. 5, pp. 83–89, Sep./Oct. 2021.

[267] Y. Wang, S. Yang, X. Ren, P. Zhao, C. Zhao, and X. Yang, "IndustEdge: A time-sensitive networking enabled edge-cloud collaborative intelligent platform for smart industry," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2386–2398, Apr. 2022.

[268] C. Jian, J. Ping, and M. Zhang, "A cloud edge-based two-level hybrid scheduling learning model in cloud manufacturing," *Int. J. Prod. Res.*, vol. 59, no. 16, pp. 4836–4850, 2021.

[269] Z. Ding, J. Wang, Y. Cheng, and C. He, "Alice: A LSTM neural network based short-term power load forecasting approach in distributed cloud-edge environment," *J. Phys. Conf.*, vol. 1624, no. 5, 2020, Art. no. 052017.

[270] T. Wang, D. Zhao, S. Cai, W. Jia, and A. Liu, "Bidirectional prediction-based underwater data collection protocol for end-edge-cloud orchestrated system," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4791–4799, Jul. 2020.

[271] C. Song, W. Xu, G. Han, P. Zeng, Z. Wang, and S. Yu, "A cloud edge collaborative intelligence method of insulator string defect detection for power IIoT," *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7510–7520, May 2021.

[272] S. Lu, X. Tang, Y. Zhu, and J. She, "A cloud-edge collaborative intelligent fault diagnosis method based on LSTM-VAE hybrid model," in *Proc. 8th IEEE Int. Conf. Cyber Security Cloud Comput. (CSCloud) Proc. 7th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, 2021, pp. 207–212.

[273] S. Maheshwari, W. Zhang, I. Seskar, Y. Zhang, and D. Raychaudhuri, "EdgeDrive: Supporting advanced driver assistance systems using mobile edge clouds networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2019, pp. 1–6.

[274] Y. Xun, J. Qin, and J. Liu, "Deep learning enhanced driving behavior evaluation based on vehicle-edge-cloud architecture," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6172–6177, Jun. 2021.

[275] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1840–1852, Mar. 2021.

[276] I. Quintana-Ramirez, L. Sequeira, and J. Ruiz-Mas, "An edge-cloud approach for video surveillance in public transport vehicles," *IEEE Latin America Trans.*, vol. 19, no. 10, pp. 1763–1771, Oct. 2021.

[277] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, and W. Zhao, "An edge computing based public vehicle system for smart transportation," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12635–12651, Nov. 2020.

[278] Q. Yuan et al., "Cross-domain resource orchestration for the edge-computing-enabled smart road," *IEEE Netw.*, vol. 34, no. 5, pp. 60–67, Sep./Oct. 2020.

[279] F. Si, Y. Han, J. Wang, and Q. Zhao, "Connectivity verification in distribution systems using smart meter voltage analytics: A cloud-edge collaboration approach," *IEEE Trans. Ind. Informat.*, vol. 17, no. 6, pp. 3929–3939, Jun. 2021.

[280] F. Song, M. Zhu, Y. Zhou, I. You, and H. Zhang, "Smart collaborative tracking for ubiquitous power IoT in edge-cloud interplay domain," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6046–6055, Jul. 2020.

[281] J. Zhong and X. Xiong, "An orderly EV charging scheduling method based on deep learning in cloud-edge collaborative environment," *Adv. Civil Eng.*, vol. 2021, Jan. 2021, Art. no. 6690610.

[282] V. Chamola, A. Sancheti, S. Chakravarty, N. Kumar, and M. Guizani, "An IoT and edge computing based framework for charge scheduling and EV selection in V2G systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 10569–10580, Oct. 2020.

[283] Y. Zhang et al., "Edge intelligence for plug-in electrical vehicle charging service," *IEEE Netw.*, vol. 35, no. 3, pp. 81–87, May/Jun. 2021.

[284] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "FedHome: Cloud-edge based Personalized federated learning for in-home health monitoring," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2818–2832, Aug. 2022.

[285] P. Pratim Ray, D. Dash, and N. Moustafa, "Streaming service provisioning in IoT-based healthcare: An integrated edge-cloud perspective," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 11, 2020, Art. no. e4109.

[286] D. Gao, C. Ju, X. Wei, Y. Liu, T. Chen, and Q. Yang, "HHHFL: Hierarchical heterogeneous horizontal federated learning for electroencephalography," 2019, *arXiv: 1909.05784*.

[287] V. K. Singh and M. H. Kolekar, "Deep learning empowered COVID-19 diagnosis using chest CT scan images for collaborative edge-cloud computing platform," *Multimedia Tools Appl.*, vol. 81, no. 1, pp. 3–30, 2022.

[288] Z. Yang, B. Liang, and W. Ji, "An intelligent end–edge–cloud architecture for visual IoT-assisted Healthcare systems," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 16779–16786, Dec. 2021.

[289] S. Ding, L. Li, Z. Li, H. Wang, and Y. Zhang, "Smart electronic gastroscope system using a cloud-edge collaborative framework," *Future Gener. Comput. Syst.*, vol. 100, pp. 395–407, Nov. 2019.

[290] D. Wu, X. Han, Z. Yang, and R. Wang, "Exploiting transfer learning for emotion recognition under cloud-edge-client collaborations," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 479–490, Feb. 2021.

[291] B. Hou and J. Zhang, "Real-time surveillance video salient object detection using collaborative cloud-edge deep reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–8.

[292] X. Wang and G. Gao, "SmartEye: An open source framework for real-time video analytics with edge-cloud collaboration," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 3767–3770.
[293] S.-W. Kim, K. Ko, H. Ko, and V. C. Leung, "Edge-network-assisted real-time object detection framework for autonomous driving," *IEEE Netw.*, vol. 35, no. 1, pp. 177–183, Jan./Feb. 2021.
[294] Z. Gao et al., "Salient object detection in the distributed cloud-edge intelligent network," *IEEE Netw.*, vol. 34, no. 2, pp. 216–224, Mar./Apr. 2020.
[295] S. Wang, S. Yang, and C. Zhao, "SurveilEdge: Real-time video query based on collaborative cloud-edge deep learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2020, pp. 2519–2528.
[296] M. Zhang, F. Wang, Y. Zhu, J. Liu, and Z. Wang, "Towards cloud-edge collaborative online video analytics with fine-grained serverless pipelines," in *Proc. 12th ACM Multimedia Syst. Conf.*, 2021, pp. 80–93.
[297] W. Zhang et al., "Deep reinforcement learning based resource management for DNN inference in industrial IoT," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7605–7618, 2021.
[298] F. Lyu et al., "Characterizing urban vehicle-to-vehicle communications for reliable safety applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2586–2602, Jun. 2020.
[299] F. Lyu et al., "Towards rear-end collision avoidance: Adaptive beaconing for connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 1248–1263, Aug. 2021.
[300] S. Duan et al., "Multitype highway mobility Analytics for efficient learning model design: A case of station traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19484–19496, Oct. 2022.
[301] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6206–6221, Jul. 2022.
[302] L. Liu, Y. Yao, R. Wang, B. Wu, and W. Shi, "Equinox: A road-side edge computing experimental platform for CAVs," in *Proc. IEEE Int. Conf. Connected Auton. Driving (MetroCAD)*, 2020, pp. 41–42.
[303] W. Wei, R. Yang, H. Gu, W. Zhao, C. Chen, and S. Wan, "Multi-objective optimization for resource allocation in vehicular cloud computing networks," *IEEE Trans. Intell. Transp. Syst.*, early access, Aug. 3, 2021, doi: 10.1109/TITS.2021.3091321.
[304] A. Adeniran, M. A. Hasnat, M. Hosseinzadeh, H. Khamfroush, and M. Rahnamay-Naeini, "Edge layer design and optimization for smart grids," in *Proc. IEEE Int. Conf. Commun. Control Comput. Technol. Smart Grids (SmartGridComm)*, 2020, pp. 1–6.
[305] Y. Zhang, X. Wang, J. He, Y. Xu, F. Zhang, and Y. Luo, "A transfer learning-based high impedance fault detection method under a cloud-edge collaboration framework," *IEEE Access*, vol. 8, pp. 165099–165110, 2020.
[306] J. Zhang et al., "An optimal dispatching strategy for charging and discharging of electric vehicles based on cloud-edge collaboration," in *Proc. IEEE 3rd Asia Energy Elect. Eng. Symp. (AEEES)*, 2021, pp. 827–832.
[307] B. Patra and K. Mohapatra, "Cloud, edge and fog computing in Healthcare," in *Intelligent and Cloud Computing*. Heidelberg, Germany: Springer, 2021, pp. 553–564.
[308] W. H. Organization. "Towards Policy for Health and Ageing." 2019. [Online]. Available: https://www.who.int/ageing/publications/alc_fs_ageing_policy.pdf,
[309] X. Tang et al., "Computing power network: The architecture of convergence of computing and networking towards 6G requirement," *China Commun.*, vol. 18, no. 2, pp. 175–185, 2021.
[310] R. Xiao-Xu, T. Jing-Tiao, D. Hui, C. Yi-Fan, C. Tiao, and Y. Xiao-Fei, "Computing power network framework based on end-edge-supercloud collaboration," *J. Comput. Appl.*, to be published.
[311] C. Telecom. "Computing Power Network for MEC." 2019. [Online]. Available: https://www.itu.int/en/ITU-T/Workshops-and-Seminars/2019101416/Documents/Xie_Yunpeng_Presentation.pdf
[312] G. Cheng, K. Chadha, and J. Duchi, "Fine-tuning is fine in federated learning," 2021, *arXiv:2108.07313*.
[313] Y. Li, F. Li, L. Chen, L. Zhu, P. Zhou, and Y. Wang, "Power of redundancy: Surplus client scheduling for federated learning against user uncertainties," *IEEE Trans. Mobile Comput.*, early access, May 26, 2022, doi: 10.1109/TMC.2022.3178167.
[314] L. U. Khan, Z. Han, D. Niyato, E. Hossain, and C. S. Hong, "Metaverse for wireless systems: Vision, enablers, architecture, and future directions," 2022, *arXiv:2207.00413*.
[315] L.-H. Lee et al., "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," 2021, *arXiv:2110.05352*.
[316] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: Survey, challenges and solutions," *J. Ambient Intell. Humanized Comput.*, to be published.
[317] U. Mattsson. "Machine Learning in Trusted Execution Environments and Quantum Computers." 2021. [Online]. Available: https://www.globalsecuritymag.com/Machine-Learning-in-Trusted,20210610,112686.html
[318] IDC. "Function Offload Coprocessors: A New Era of Decentralized and Distributed Computing." 2020. [Online]. Available: https://www.globalsecuritymag.com/Machine-Learning-in-Trusted,20210610,112686.html

Sijing Duan (Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University, Changsha, China. Her research interests include Internet of Things, edge computing, big data measurement, and data-driven application design.

Dan Wang (Student Member, IEEE) received the B.E. degree in electronic engineering and the M.Sc. degree in control science and engineering from China Agricultural University, China, in 2015 and 2017, respectively. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Central South University. Her research interest focuses on privacy computing and edge computing.

Ju Ren (Senior Member, IEEE) received the B.Sc., M.Sc., Ph.D. degrees in computer science from Central South University, China, in 2009, 2012, and 2016, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University, China. His research interests include Internet of Things, edge computing, edge intelligence, as well as security and privacy. He received many Best Paper Awards from the IEEE Flagship Conferences, including IEEE ICC'19 and IEEE HPCC'19, the IEEE TCSC Early Career Researcher Award in 2019, and the IEEE ComSoc Asia-Pacific Best Young Researcher Award in 2021. He was recognized as a Highly Cited Researcher by Clarivate (2020–2022). He currently serves as an Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and *Peer-to-Peer Networking and Applications*. He also served as the General Co-Chair for IEEE BigDataSE'20, the TPC Co-Chair for IEEE BigDataSE'19, the Publicity Co-Chair for IEEE ICDCS'22, the Poster Co-Chair for IEEE MASS'18, the Symposium Co-Chair for IEEE/CIC ICCC'23&19, IEEE I-SPAN'18, and VTC'17 Fall.

Feng Lyu (Senior Member, IEEE) received the B.S. degree in software engineering from Central South University, Changsha, China, in 2013, and the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018. From September 2018 to December 2019 and from October 2016 to October 2017, he worked as a Postdoctoral Fellow and was a visiting Ph.D. student with the BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is currently a Professor with the School of Computer Science and Engineering, Central South University, Changsha, China. His research interests include vehicular networks, beyond 5G networks, big data measurement and application design, and edge computing. He is the recipient of the Best Paper Award of IEEE ICC 2019. He currently serves as an Associate Editor for IEEE SYSTEMS JOURNAL and a Leading Guest Editor for *Peer-to-Peer Networking and Applications*, and served as a TPC member for many international conferences. He is a member of the IEEE Computer Society, Communication Society, and Vehicular Technology Society.

Ye Zhang (Member, IEEE) received the B.Sc. degree in computer science and technology from Beihang University in 2007, and the M.E. degree in software engineering from Peking University in 2011. She is currently a Teacher with the School of Computer Science, Beijing Information Science and Technology University. Her interests include edge computing and natural language processing.

Huaqing Wu (Member, IEEE) received the B.E. and M.E. degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014 and 2017, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2021. She worked as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, MacMaster University from 2021 to 2022. She is currently an Assistant Professor with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB, Canada. Her current research interests include B5G/6G, space-air-ground integrated networks, Internet of vehicles, edge computing/caching, and artificial intelligence for future networking. She received the Best Paper Award at IEEE GLOBECOM 2018 and Chinese Journal on Internet of Things 2020, and the prestigious Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship Award in 2021

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a Registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society (ComSoc), and the Technical Recognition Award from Wireless Communications Technical Committee in 2019, and the AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE GLOBECOM'16, IEEE INFOCOM'14, and IEEE VTC'10 Fall, IEEE GLOBECOM'07 and the Chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the President Elect of the IEEE ComSoc. He was the Vice President for Technical and Educational Activities, the Vice President for Publications, the Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a Member of IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*.