A Survey of Recent Advances in Edge-Computing-Powered Artificial Intelligence of Things

Zhuoqing Chang, *Student Member, IEEE*, Shubo Liu, Xingxing Xiong, Zhaohui Cai, and Guoqing Tu

***Abstract*—The Internet of Things (IoT) has created a ubiquitously connected world powered by a multitude of wired and wireless sensors generating a variety of heterogeneous data over time in a myriad of fields and applications. To extract complete information from these data, advanced artificial intelligence (AI) technology, especially deep learning (DL), has proved successful in facilitating data analytics, future prediction and decision making. The collective integration of AI and the IoT has greatly promoted the rapid development of AI-of-Things (AIoT) systems that analyze and respond to external stimuli more intelligently without involvement by humans. However, it is challenging or infeasible to process massive amounts of data in the cloud due to the destructive impact of the volume, velocity, and veracity of data and fatal transmission latency on networking infrastructures. These critical challenges can be adequately addressed by introducing edge computing. This article conducts an extensive survey of an end-edge-cloud orchestrated architecture for flexible AIoT systems. Specifically, it begins with articulating fundamental concepts including the IoT, AI and edge computing. Guided by these concepts, it explores the general AIoT architecture, presents a practical AIoT example to illustrate how AI can be applied in real-world applications and summarizes promising AIoT applications. Then, the emerging technologies for AI models regarding inference and training at the edge of the network are reviewed. Finally, the open challenges and future directions in this promising area are outlined.**

***Index Terms*—Artificial intelligence (AI), deep learning (DL), edge computing, Internet of Things (IoT), machine learning (ML).**

I. Introduction

BENEFITTING from the extensive use of the Internet and the rapid development of many wired and wireless connected devices, the Internet of Things (IoT) matures promptly and plays an increasingly significant role in every aspect of life by providing many crucial services, such as information exchange and monitoring [1]. With the extensive applications of IoTs, the total installed IoT-based devices is projected to amount to approximately 41.6 billion, and nearly 79.4 Zettabytes (ZBs) of data may be generated and consumed in 2025 [2]. Thus, some nonnegligible challenges that the IoT faces are explosive data generation and reliable data collection between heterogeneous devices in a wide range of applications covering various backgrounds and requests. Cloud computing plays a crucial role in IoT systems, where the vast resources available in the cloud can provide ubiquitous on-demand computing and storage capabilities to support these devices [3]. Additionally, these data may consist of multimedia information, from images, sounds, and videos to structured data (e.g., temperature and humidity). Advanced tools are needed to glean insights from a large volume of raw data. Facilitated by the recent achievements of algorithms, computing capabilities and big data processing necessities, artificial intelligence (AI), especially its essential sector of deep learning (DL), has achieved unprecedented success in data analysis, future prediction and decision making [4]. Clearly, the AI of Things (AIoT), an integrative technology combining both AI and the IoT, is starting to garner its share of the spotlight with the support of cloud centers. In the AIoT era, large amounts of data generated by IoT devices provide perfect opportunities for training AI models to reliably mine valuable data from a noisy and complex environment for intelligent analysis and decision making.

The cloud-centric AIoT requires the massive amount of heterogeneous data collected from IoT sensors to be transmitted to the cloud center through a wide-area network (WAN) for further processing and analysis before delivering the feedback to end devices [5]. Although the cloud center an has unlimited computational capacity, such a cloud-based AIoT architecture is ill-suited for time-critical and privacy-sensitive applications due to the great pressure on network bandwidth, the inherent latency constraints of network communication and the potential to expose private and sensitive information during data offloading and remote processing [6]–[8]. Edge computing seems to be a promising technique to remedy these issues, which brings computational resources closer to the data source with a relatively light access burden and a low transmission delay [9]. It is extremely suitable for the AIoT because AI models, especially DL models, that depend greatly on computation and storage resources can still work fluently and cooperatively by partitioning the layers into several parts and offloading the computation-intensive tasks to edge servers [3]. Such a computing paradigm coupled with AI can assist users better and more intelligently, where AI models function as

Manuscript received October 7, 2020; revised December 21, 2020, February 20, 2021, April 4, 2021, and May 10, 2021; accepted June 9, 2021. Date of publication June 14, 2021; date of current version September 6, 2021. This work was supported by the Major Projects of Technical Innovation of Hubei Province under Grant 2018AAA046. *(Corresponding author: Shubo Liu.)*

Zhuoqing Chang, Shubo Liu, Xingxing Xiong, and Zhaohui Cai are with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: changzhuoqing@whu.edu.cn; liu.shubo@whu.edu.cn; xiong_xx@whu.edu.cn; zhcai@whu.edu.cn).

Guoqing Tu is with the School of National Cybersecurity, Wuhan University, Wuhan 430072, China (e-mail: tugq2000@163.com).

Digital Object Identifier 10.1109/JIOT.2021.3088875

a powerful tool to mine valuable information from raw data, make real-time decisions [10] and to dynamically manage various resources of the edge platforms [11], [12]. Rather than transmit all the raw data to the cloud for overall analysis, edge computing-assisted AIoT solutions essentially enable AI models to work in the field. These solutions can lighten the burden of data transmission through the network backhaul, further reduce the cost of network processing and maintenance and make timely decisions by positioning computational capabilities near end devices [13]. Additionally, these methods protect sensitive data from being abused by illegal operators or hijacked by attackers [14].

This article investigates convergence of AI and the IoT from the perspective of end-edge-cloud collaboration, where immediate and real-time responses are enabled by using AI capacities for processing raw data at end or edge devices and higher accuracy results are gained by using cloud analytics in collaboration. The AIoT can bring numerous benefits to human beings in a spectrum of domains and form a ubiquitous intelligent collaborative environment; however, significant challenges must be overcome before fully realizing the potential of AIoT. Thus, this article aims to present a comprehensive survey of recent advances, open challenges and future directions for the AIoT.

A. Related Work

Several published references present AI and machine learning (ML) or DL methods that have been used in the domain of the IoT. For example, Ghosh *et al.* [15] proposed that integrating the IoT with AI is much more than a way of making human life easier; additionally, this integration brings security concerns and ethical issues. Din *et al.* [16] reported different IoT-based ML mechanisms in healthcare, smart grids, and vehicular communications and describes the basic aim of ML applied to the IoT. To lower the latency and bandwidth of data transmission, edge computing is also investigated in the context of the IoT. Deng *et al.* [7] demonstrated that AI can not only endow edges with greater intelligence and optimality but also help run AI models at edges. Cui *et al.* [17] described the potential of ML methods for traffic profiling, device identification, system security, IoT applications, edge computing and software-defined networking (SDN). In addition to reviewing the characteristics of IoT data, state-of-the-art ML and DL methods and IoT applications using different deep neural network (DNN) models, Mohammadi *et al.* [18] reviewed approaches and technologies for running DL models on resource-constrained devices and edge servers. Efforts to devise compression and acceleration techniques that help deploy DL algorithms on resource-hungry mobile and embedded devices are further differentiated and discussed in [19] to better satisfy the requirements of real-time applications and user privacy protection. A novel offloading strategy is essential to advance the implementation of DL-based applications in IoT systems aided by edge computing and cloud [20]. Chen and Ran [14] and Zhou *et al.* [21] aimed recent and in-depth research of relevant works that deal with AI inference and training at the network edge.

AI helps more effectively handle seemingly insurmountable resource allocation challenges in edge computing scenario. Sodhro *et al.* [22] proposed a forward central dynamic and available approach to managing the running time of sensing and transmission processes in AI-enabled IoT devices for industrial platforms. Dai *et al.* [12] employed a novel deep reinforcement learning (DRL) approach for intelligently orchestrating edge computing and caching resources to maximize the system utility for vehicular networks. Survey [23] focuses on the combination of AI and edge computing in the field of the Internet of Vehicles (IoV), where AI is used mainly for resource allocation, computational task scheduling, and vehicle trajectory prediction in dynamic environments.

Surprisingly little work has been done to put forward a general concept and an architecture for the AIoT. The existing works on IoT-based AI implementation rely on cloud platforms; however, this approach is unacceptable for delay-sensitive services. Edge computing extends cloud analytics to the network edge and covers its shortage. In contrast to other surveys, this article explores the combination of the IoT with AI aided by edge computing and the cloud.

B. Contributions of This Survey

By referring to previous works, this article provides a conceptual introduction to IoT, AI and edge computing technologies. The main goal is to conduct a comprehensive analysis on the potentials of integration of the IoT and AI technologies with the assistance of edge computing coordinated by the cloud. This article put emphasis on seven representative AIoT application scenarios and is especially interested in techniques enabling efficient and effective deployment of AI models in an end-edge-cloud cooperation mood. In summary, the contributions of this article can be summarized as follows.

1) An overview of fundamental technologies supporting the AIoT are given in terms of the general architecture of the IoT, state-of-the-art AI methods accompanied by key characteristics, and edge computing-related paradigms along with corresponding hardware and systems.
2) Confluence of AI and the IoT is the core of this article. In this respect, benefits of incorporating AI into IoT systems are first illustrated. An end-edge-cloud collaborative architecture of AIoT are then proposed. A practical example of AIoT applications is additionally given to further illustrate how AI can be applied in real-world applications.
3) Some promising applications of AIoT are surveyed in a variety of domains, such as the IoV, smart healthcare, smart industry, smart homes, smart agriculture, smart grids and smart environment.
4) The recent approaches and technologies for performing AI inference on an AIoT hierarchy from resource-hungry end devices to edge servers and the cloud are summarized.
5) The enabling technologies for decentralized AI training among various end devices and edge servers of an AIoT hierarchy are discussed.

6) The open challenges and future directions for constructively and fruitfully merging of AI and the IoT are outlined.

C. Outline of This Survey

The remainder of this article is arranged as follows. Section II presents the fundamentals related to end-edge-cloud collaborative AIoT systems in terms of the IoT architecture, basic AI models with an emphasis on ML and DL models, and edge computing. Section III describes the benefits of the convergence of AI and the IoT, highlights the AIoT architecture and discusses an AIoT example to further illustrate how AI can be applied in real-world applications. AIoT applications in different domains (e.g., the IoV, smart healthcare, smart industry, smart homes, smart agriculture, smart grids, and smart environment) are surveyed in Section IV. Section V investigates the enabling technologies for AI inference from the end-edge-cloud orchestrated perspective, including inference on-device and coinference at the edge, and private inference technologies. Section VI taxonomically summarizes the enabling technologies for distributed model training in AIoT, focusing on methods of decentralized AI training at the network edge, model training updates and security enhancement. In Section VII, some open research challenges and future directions for the AIoT are envisioned for fostering continuous research efforts. Finally, Section VIII concludes this article.

II. Fundamentals of Artificial Intelligence of Things

This section reviews the general architecture of the IoT in brief, then presents basic AI models with an emphasis on ML and DL models and additionally gives an overview of edge computing in terms of related paradigms, relationship with the fifth generation (5G), hardware, and systems.

A. Introduction to the Internet of Things

With the aid of sensors, wired and wireless networks and cloud computing, the IoT achieves a comprehensive perception and a ubiquitously connected environment. Although there is no unified definition of the IoT, a typical IoT architecture is largely recognized, as shown in Fig. 1. Generally, the IoT architecture is composed of three layers, namely, the perception layer, network layer and application layer.

1) Perception Layer: The perception layer provides the core capability enabling the comprehensive awareness of the environment; this layer includes such diverse devices and technologies as sensors, actuators, radio-frequency identification (RFID), 2-D codes, and multimedia information collection devices. These devices are used mainly to sense and collect physical data, and data is generally produced in trillions of bytes with a variety of attributes, including various physical quantities, identity signs, location information, and audio and video data. Additionally, these devices can respond to the environment.

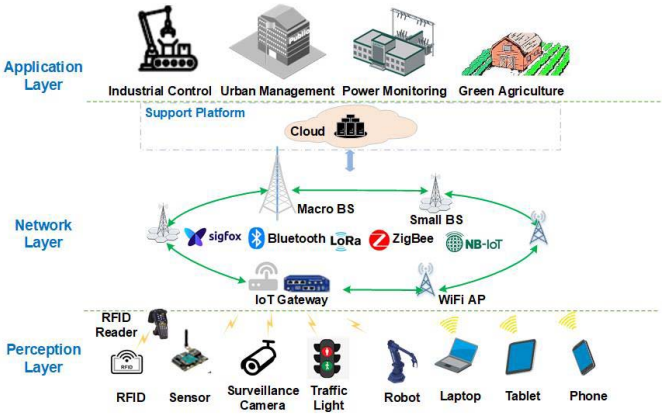

Fig. 1. General architecture of the IoT.

2) Network Layer: The network layer is the most standardized among the three layers of the IoT, in which the devices in the perception layer can communicate with IoT gateways, wireless-fidelity (Wi-Fi) access points (APs) and base stations (BSs) to transmit data to other parts quickly, accurately and safely. This layer enables a device to communicate at a short-range to long-range distance by using a variety of communication protocols, such as wired and wireless networks, including Bluetooth, ZigBee, Sigfox, long-range radio (LoRa), and Narrowband IoT (NB-IoT). The data generated in the perception layer need to be transmitted to the server via the network layer promptly and accurately.

3) Application Layer: The application layer is equivalent to the control layer and decision-making layer of the IoT, in which a mass of polymorphic and heterogeneous data with rich semantics are analyzed. The application layer can provide a myriad of applications, such as industrial control, urban management, power monitoring, and green agriculture.

B. Basics of Artificial Intelligence

From Siri to self-driving vehicles, AI developed rapidly, is adopted in a wide range of applications and corroborates its outstanding performance. Algorithms exert a crucial function in AI, where simple algorithms can be applicable to simple applications, while more complex ones can build strong AI. Typically, ML, a subset of AI, is the most mainstream method used by systems to automatically learn from the data, identify patterns and make decisions from experience without human intervention or assistance. There are a variety of ML models, ranging from basic algorithms to highly complex ones, such as support vector machines (SVMs), decision trees (DTs), k-means clustering, and neural networks. DL, a unique branch of ML, is quite different from classic ML algorithms, which uses a hierarchical neural network to make the model more complex and enables automatic learning by absorbing a great deal of unstructured data, such as images, sound, text and video. Many DL models, as exemplified by multilayer perceptrons (MLPs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and generative adversarial networks (GANs), have been developed to train neural

networks to make classifications and predictions. This section will introduce several traditional ML algorithms and present some typical neural network algorithms accompanied with the corresponding functions, including a distinctive reinforcement learning (RL) method.

1) Traditional Machine Learning: This section will deliver a brief introduction on several traditional ML algorithms, such as SVMs, DTs, and *k*-means clustering.

a) Support vector machine: SVMs are a kind of generalized linear classifier that classifies data by supervised learning. SVMs can also perform nonlinear classification by using the kernel method. The learning policy of an SVM is to maximize the separation between data points, which is formalized as a convex quadratic programming problem. SVMs have been successfully applied in pattern recognition problems, such as in text classification [24] and image recognition [25].

b) Decision tree: DTs are classification algorithms that predict the labels of data by iterating the input data via a learning tree, the main advantages of which are readability, a fast classification speed and understandability. However, applications of DTs are confined to linear separable data. DTs are constructed by selecting the partition method with the largest reduction in the entropy of the data set. DTs can be used in classification [26], medical diagnoses [27], and so on.

c) k-means clustering: *k*-means cluster analysis algorithms are based on the iterative solutions. With advantages of linear complexity and simple realization, *k*-means cluster is employed to solve node clustering problems [28]. Given a set of data points and the required number of clusters *k*, where *k* is specified by the user, the *k*-means algorithm divides the data into k clusters repeatedly according to a certain distance function. *k*-means clustering performs well in grouping the same things from a randomly distributed set of things, such as in object detection [29].

2) Neural Networks: With various types of DNNs, DL models are capable of extracting accurate information from raw sensor data collected by IoT devices and make contributions to more complicated tasks ranging from image classification and retrieval to natural language processing. It cannot be ignored that DL models can still work in edge computing environment due to their multilayer structure. Therefore, this section briefly introduces some DNN models in terms of their network structures and functions.

a) Multilayer perceptron: MLPs are a type of feedforward artificial neural network (ANN), as shown in Fig. 2(a). Adopting the fully connected neural network (FCNN) technique, the input of a cell is fed forward to a hidden cell, then activated by cells belonging to the following layer, and is finally transmitted to the output cell. By calculating the error between the output layer and the previous hidden layer, the error can be reduced by adjusting the internal weights between each pair of layers. MLPs are more accurate than other techniques in healthcare applications [30]. Moreover, MLP-based models for blind entity identification are explored to help estimate the wireless link quality of a network, which are successfully implemented in agricultural IoT [31].

b) Convolutional neural network: CNNs can capture the correlations between adjacent data blocks and extract high-level features by using convolutional and pooling operations, as depicted in Fig. 2(b). Unlike FCNNs, CNNs can extract features while reducing the number of parameters and the computation time by pooling operations to reduce the chance of overfitting [32]. The rectified linear unit (ReLU) activation function can accelerate training time without affecting the generalization of the network. These characteristics make CNNs excellent at analyzing not only images and speech signals but also structural data similar to images [33], [34].

c) Recurrent neural network: RNNs are developed to handle sequential (speech or text) or time-series (sensor data) inputs of different lengths and can estimate energy consumption in smart grids and so on. As the typical structure of RNNs depicted in Fig. 2(c), the input of each neuron consists of information from the upper layer and information from its own previous channel [4]. The neuron is also furnished with a feedback loop that provides the current output for the next step as the input. RNNs are ideal candidates to predict future information and restore missing parts of sequential data [35], [36].

d) Long short-term memory: LSTMs are considered extensions of RNNs. In contrast to RNNs, LSTM units employ a gate structure and a well-defined memory cell to actively control unit states, shown in Fig. 2(d), where gradient explosion is solved by controlling (prohibiting or allowing) the flow of information. The gates exploit *sigmoid* or *tanh* as their activation function. Vanishing gradients caused by using these activation functions during the training of other models do not take place in LSTMs because the computations stored in the memory cells are not distorted over time. LSTMs outperform RNNs when dealing with data featuring a long dependency over time. LSTMs have been investigated for long dependencies in IoT applications, such as pedestrian trajectory prediction [37] and traffic flow prediction [38].

e) Generative adversarial network: GANs, originating from game theory, consist of two neural networks, namely, a generator and discriminator, as illustrated in Fig. 2(e). The generator aims to generate new data after learning the data distribution, while the discriminator decides whether the input data come from the real data or the generator. Both networks constantly optimize their abilities to generate and distinguish data in the adversarial process until a Nash equilibrium is found [39]. In IoT applications, GANs can be used to generate new data beyond the available data. In multiresident activity recognition in smart homes, GANs can be helpful in creating more data to train DL models [40].

3) Reinforcement Learning: RL, another undeniably important DL method, allows various software agents and machines to take suitable actions by interacting with their environment and aimed at discovering mistakes and maximizing rewards in a particular situation. Generally, the interaction between the agent's state and actions through the environment is described as the Markov decision process (MDP). DRL is a type of RL that combines DL and RL where DL uses DNNs to fit the *Q*-value function, thereby addressing the explosion of state-action space challenges. DRL has wide utilization in recommendation [41], wireless communication [42], resource allocation [43], and so on. There are two typical kinds of DRL

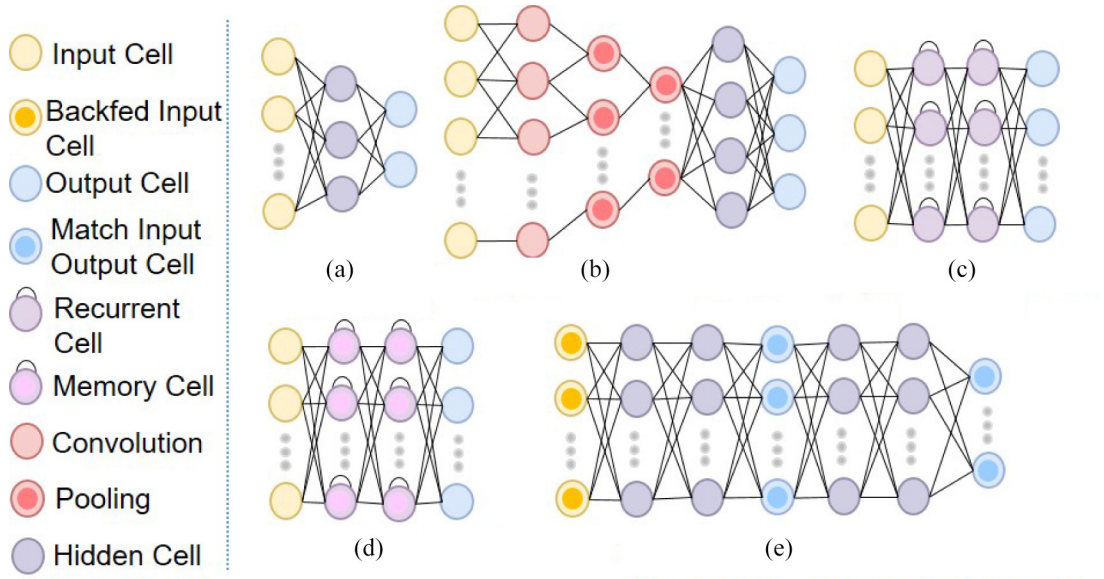

Fig. 2. Basic structures of typical DL. (a) Multilayer perception. (b) CNN. (c) RNN. (d) Long short-term memory. (e) GAN.

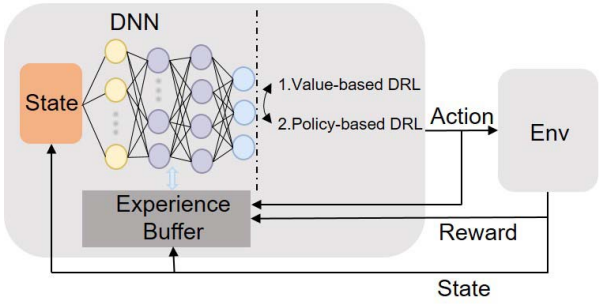

Fig. 3. Architecture of typical DRL.

algorithms, namely, value-based DRL and policy-based DRL, as shown in Fig. 3.

a) Value-based deep Q-learning: Deep Q-learning (DQL) is a typical representative of value-based DRL, where DNNs are employed to fit the Q-values of actions. Experience replay is introduced to fix the instability problem of value functions represented by nonlinear network and nonstatic distribution problems. However, the max operation is employed in a deep Q-network to select and measure actions, which may result in overestimating the action values. Double DQL (DDQL) decouples selection and evaluation to address this problem [44].

b) Policy-based deep Q-learning: Policy-based DRL can handle a continuous action space with better convergence; in this method, the policy parameters are updated by continuously calculating the gradient of the policy expectation reward. In such situations, DNNs are employed to parameterize the policy and are then optimized by the policy gradient method. The actor–critic (AC) [45] framework is widely applied in policy gradient-based DRL, where the Actor selects actions and modifies the policy distribution based on the score from the Critic, while the Critic learns a value function and estimates the performance of the policy updated by the Actor. Other strategy optimization methods include the deep deterministic policy gradient (DDPG) [46] and proximate policy optimization (PPO) [47].

C. Overview of Edge Computing

Cloud computing refers to the concept of delivering different services on demand by using networking, virtualization, distributed computing, utility computing and software services. Cloud computing is a service-oriented architecture that provides flexible services, reduces the information technology overhead for end users, and the total cost of ownership [48]. However, the cloud center is usually built in a remote area far from end users, thus possibly causing data transmission delays. With a soaring number of IoT devices, the cloud cannot satisfy the requirements of latency-sensitive and privacy-critical applications [21]. The emergence of edge computing is aimed at migrating computational tasks to edge devices near sensors and actuators, which can alleviate the pressure of data transmission, reduce end-to-end latency, and thus enable real-time services. A diversity of devices can serve as edge computing platforms: switches, cellular BSs or IoT gateways [9], which makes edge computing flexible and scalable deploy various services at anywhere between end users and the cloud. Typically, edge computing is considered an extension of the cloud platform and works independently and effectively in some scenarios or collaborates with the cloud platform. This section reviews edge computing-related paradigms, the relationship between 5G and edge computing, and edge computing hardware and systems.

1) Edge Computing-Related Paradigms: Many emerging technologies have been proposed to work at the edge of the network; these technologies have similar but distinct paradigms, including mobile cloud computing (MCC) [49], cloudlets [50], fog computing [51], and multiaccess edge computing (MEC) [8]. The term "edge computing" represents this

set of burgeoning technologies. Below, some typical concepts related to edge computing are discussed and differentiated.

a) Cloudlet: Cloudlets, proposed by Carnegie Mellon University, are envisioned as a mobility-enhanced data centers with certain computation and storage capabilities and located near mobile devices. The cloudlet is considered as the middle layer of a three-tier architecture: 1) mobile devices; 2) the microcloud; and 3) the cloud. Cloudlets have the features of computing, connectivity and security capabilities, proximity to mobile users, virtualization management technologies and implementation of related functionality based on standard cloud technologies [50], [52]. Compared with other paradigms, cloudlets are more similar to mobile clouds and closer to mobile devices, which are suitable for real-time and mobile scenarios.

b) Fog computing: Fog computing, proposed initially by Cisco, is considered an effective extension of and supplement to traditional cloud computing, which places resources and services (computation, storage, networking, and processing) on a path from end devices to the cloud [51]. Fog nodes, such as APs, gateways, and BSs, can be deployed on any infrastructure in a specific geographic area [53]. Fog computing provides real-time collaborative services with less latency for numerous interconnected IoT devices via distributed fog nodes [54]. Unlike cloudlets and MEC, fog computing is focused more on the IoT and the end side.

c) Mobile-edge computing: MEC, standardized by the European Telecommunications Standards Institute (ETSI), provides services by allocating computation and storage resources at the edge of cellular networks, e.g., wireless BSs [9]. BSs are the major access gates for IoT devices, and MEC enables users to deploy services flexibly and directly by using only one hop. Later, MEC is extended by ETSI to MEC by supporting more wireless communication technologies, such as Wi-Fi. MEC is extensively used in autonomous driving (AD) vehicles, wearable devices, and so on.

2) Fifth-Generation Mobile Networks and Edge Computing: 5G technology standard is seen as the most promising wireless cellular network standard to cater to the requirements of next-generation networks. Many ultradense edge devices will be deployed in 5G systems, including mainly small cell BSs and wireless APs. These devices are often equipped with certain computing and storage abilities, thus enabling ubiquitous mobile computing. In contrast to the main goal of the first generation (1G) to the fourth generation (4G), aimed at higher wireless speeds to support the transition from voice-centric to multimedia-centric traffic, the goal of 5G is to provide services for the explosive evolution of information communication technology and all kinds of interactive applications with latency requirements of less than 10 ms or even 1 ms for specific situations [55]. IoT devices with limited resources for computing, communication and storage have to execute all tasks with the aid of cloud centers. However, the ambitious millisecond-scale latency services in 5G cannot be realized with only the support of the cloud. The volume, velocity, and veracity of the data will heavily burden the network bandwidth and the center cloud. MEC, a viable solution, adopts a decentralized model that allocates computing and storage resources closer to end users, which is aligned with the concept of a next-generation network in which tasks are generated and executed locally [56]. Edge computing and 5G are closely bound to realize high bandwidth and real-time interactive AIoT services, where 5G enables mass connections, while edge computing provides low latency services by providing computing and storage capabilities near the data source.

3) Edge Computing Hardware and Systems: This section introduces some hardware supporting the execution of AI algorithms for both end devices and edge nodes and presents edge-cloud systems for AIoT applications (summarized in Table I).

a) Hardware for edge devices: Edge devices designed for executing AI models can generally be classified by their technical architecture into four types.

1) *Application-Specific Integrated Circuit (ASICs) Chip:* An application-specific integrated circuits (ASICs), integrated circuits, favor a specific application rather than general functions. Because of their small size, low power consumption and good security and performance, ASICs can meet the demand of edge-computing patterns for AI algorithms. The DianNao family [58], consisting of DianNao, DaDianNao, ShiDianNao, and PuDianNao, is a set of DNN accelerators that uses efficient memory access to minimize latency and energy consumption. Unlike the other DianNao family accelerators, ShiDianNao [74] is an embedded device for image applications. Edge tensor processing units (TPUs) [57] are Google's custom-developed design used to facilitate ML workloads. Edge TPUs achieve high performance in terms of their small physical area and low energy consumption.
2) *Graphics Processing Unit-Based Product:* Graphics Processing Units rely on the inherent data parallelism of a mining program to improve the actual throughput, thereby achieving a higher speed than central processing units (CPUs). This characteristic makes GPUs suitable for implementing AI algorithms. Thus, it is a good option to design an edge device equipped with a GPU. Jetson TX1, TX2 [75], and DRIVE PX2 [76], manufactured by NVIDIA, are embedded AI computing devices with a small size, low latency and high-power efficiency.
3) *Field-Programmable Gate Array (FPGA)-Based Device:* FPGAs are highly flexible programmable hardware with low energy, parallel computing resources and high security, on which developers familiar with the hardware description language can quickly execute AI algorithms. However, FPGAs have worse compatibility and more limited programming capabilities than GPUs. Xilinx ZYNQ7000 is a commonly used FPGA-based AI accelerator [77].
4) *Brain-Inspired Chip:* Brain-inspired chips adopt a neuromorphic architecture, implementing programmable neurons by using silicon technology and processing highly sophisticated tasks by mimicking the human brain with synapses. Brain-inspired chips support dramatically accelerated processing of neural network applications in real time with extremely low energy consumption. IBM

TABLE I
SUMMARY OF THE SELECTED EDGE DEVICES AND EDGE COMPUTING SYSTEMS

	Productions	Owners	Features/Targets
Edge devices	TPU [57]	Google	• Accelerates speed and reduces power consumption with a slight accuracy loss
	DianNao family [58]	Cambrian	• Hardware accelerators that minimize memory transfers
	Turing GPUs [59]	NVIDIA Corporation	• Processors process massive data in parallel at a high speed but with high energy consumption
	7 Series FPGA [60]	Xilinx	• Achieves high-performance computing with low energy consumption and high flexibility
	HiSilicon Ascend series [61]	Huawei	• Scenarios are extended from the data center to the edge and devices with a significantly improved energy efficiency ratio
	Exynos 9820 [62]	Samsung	• Executes AI tasks for the mobile future via tricluster CPU
	Xeon D-2100 [63]	Intel	• Supports optimized solutions for space- and power-constrained operational networks, storage, and cloud edges
	TrueNorth [64]	IBM	• Completes all kinds of learning tasks quickly with extremely low power consumption
Edge computing systems	CORD [65]	The Open Network Foundation	• Delivers a cloud-native and programmable platform to build an operational edge datacenter with built-in service capabilities for network operators
	EdgeX [66]	Linux Foundation	• Provides users with interoperable management for many sensors or devices across industrial fields
	Akraino Edge Stack [67]	Linux Foundation	• Integrated edge cloud platform based on a source software stack
	Azure IoT Edge [68]	Microsoft	• Supplies distributed edge management with zero-touch provisioning of IoT devices
	AWS IoT Greengrass [69]	Amazon	• Allows edge devices to perform local operations and intermittently communicate with the cloud and other devices
	KubeEdge [70]	Huawei	• Native support for the collaboration of the cloud and edge
	OpenEdge [71]	Baidu	• Shield computing framework that simplifies application production and is deployed on demand
	OpenVDAP [72]	Connected and Autonomous dRiving Lab	• An open full-stack edge-based platform for real-field vehicular data analysis
	VideoEdge [73]	Microsoft Research	• A video stream analytic system that seeks the best tradeoff between multiple resources and accuracy

TrueNorth [78] and Intel Loihi [79] are typical neuromorphic processor chips that can be applied well to complex AI algorithms.

b) Edge computing systems: Techniques for edge computing systems are blooming. Researchers focus on constructing high-performance edge computing systems with a microservice architecture to fit the complex configurations and meet the resource-constrained demands of AI algorithms. Azure IoT Edge [68], released by Microsoft, is a hybrid edge-cloud system that migrates an application from the cloud to the edge. EdgeX Foundry [66] and Apache Edgent focus on IoT edge applications and are deployed on a local area network (LAN). EdgeX Foundry, launched by the Linux Foundation, concentrates on controlling many end devices, while Apache Edgent [80], published by the Apache Software Foundation, aims to accelerate data analysis. In addition, Central Office Re-architected as a Datacenter (CORD) [65] and Akraino Edge Stack [67] can be deployed in a telecom infrastructure to support mobile-edge services, which are suitable for AD and drone applications.

III. ARTIFICIAL INTELLIGENCE OF THINGS

IoT systems have built a ubiquitously connected world, where IoT devices collect millions of data sets and perform no analysis. However, many practical services require analytical techniques to initiate appropriate actions. To address this issue, AI is introduced into the IoT, heralding the era of AIoT and achieving ubiquitous intelligent collaboration. Powered by AI, AIoT enables devices to perform self-driven analytics and make smart decisions with minimal human intervention. In this section, opportunities to fuse AI and the IoT, the general AIoT architecture with the support of edge computing and the cloud, and a practical example of AIoT applications will be reviewed.

A. Opportunities for Integrating Artificial Intelligence With the Internet of Things

The data generated by IoT devices have many properties, namely, polymorphism, heterogeneity, timeliness, accuracy, massive-scale and rich semantics. Real-time data for all events must be processed promptly. AI can effectively and efficiently mine valuable information from these data and make decisions. Moreover, AI models can be deployed on every layer of IoT systems with enhanced performance. The synergy of AI and IoT is named AIoT, benefits of which are illustrated as follows.

1) High Flexibility: Generally, the architecture of AIoT is highly flexible. In AIoT systems, end devices equipped with certain computational and storage capacities and edge servers are also utilized with the support of the cloud. More complicated Al models can be implemented on a device, edge servers and the cloud in a cooperative mode. Different Quality of Service (QoS) guarantees can be provided according to requirements, such as services in different delay ranges and predictions with different accuracy requirements. Additionally, unintelligent IoT devices can make decisions and predictions by deploying AI models in the cloud.

2) Enhanced Interactivity: The AIoT also addresses the concept of geographically distributed aspects and provides more opportunities for more end devices to participate, thereby significantly enhancing the interactivity of devices. End devices are often of various types, such as sensors and cameras. Small-size AI tasks can directly work on end devices to make real-time decisions. For large-size AI tasks that require more computing resources, the execution of AI models needs cooperation between end devices and edge servers. Taking AD as an example, the sensors need to interact with each other, and the heterogeneous data generated by different types of sensors require data fusion to make accurate predictions. The actuators work together to control the movement of the vehicle according to the prediction results.

3) Intelligent Decisions and Accurate Predictions: It is difficult to extract valuable information from the large-scale data collected by millions of IoT devices. AI-based algorithms are suitable for intelligently collecting data into a meaningful group and mining valuable information. AIoT combines data collected by IoT devices with intelligence offered by AI models, where AI models conduct automatic learning with seamless data, observe the environment, and make predictions and decisions with minimal human intervention.

4) Various Applications: IoT devices have a variety of types, such as cameras, temperature sensors, glucose sensors, and radar. These devices can sense and actuate the physical world. Additionally, AI algorithms can handle different types of data sets. Thus, a variety of applications, ranging from smart healthcare to the IoV and smart industry, can be realized by AIoT systems.

B. General Architecture of Artificial Intelligence of Things

In this section, an overview of the AIoT architecture is illustrated, which adopts a tri tier architecture similar to the architecture of the IoT, as shown in Fig. 4. From bottom to top are the end layer, edge layer and cloud layer, and the cloud layer can coordinate the end layer and the edge layer. The end layer can proprocess or analyze data on premises and make early decisions. The proprocessed data from the end layer can be aggregated in the edge or cloud layer for deep disposal. Each layer is further described as follows.

1) End Layer: The end layer in the AIoT system functions similarly to the perception layer in the IoT, where millions of interconnected sensors and actuators are deployed in a wide area. Unlike the perception layer in the IoT, the end layer not only is responsible for sensing, actuating and controlling the physical world but also executes small AI computational tasks or preprocesses data. Thus, the devices, ranging from smartphones to Raspberry Pi Bare Bones devices, typically have certain computing and storage capacities. The preprocessed compact structured information can reduce the latency, network bandwidth and cost of data transmission, which is of vital importance to AIoT applications. One obstacle is that only small-size AI models can be implemented on resource-hungry end devices. Thus, the emerging technologies designed for compressing neural networks have elicited escalating the attention of researchers, as reviewed in Section V-A. Moreover,

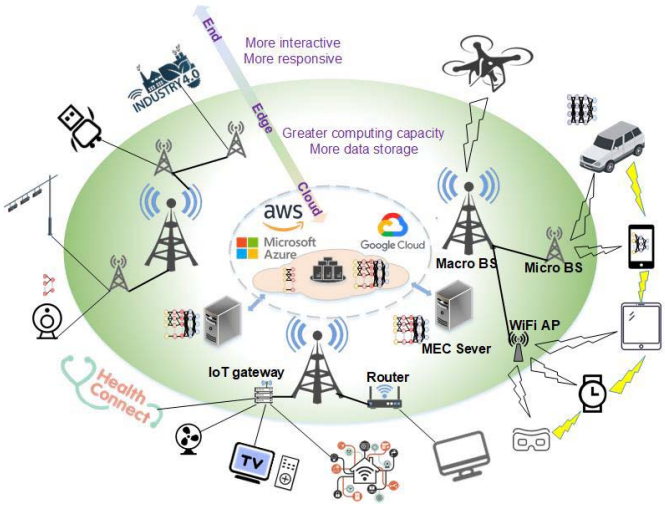

Fig. 4. Overview of an AIoT architecture.

these end devices can exchange data with each other or transmit data to the edge nodes via wireless protocols for further processing.

2) Edge Layer: The network in the edge layer is often empowered by certain computation and storage capabilities. The edge layer usually has various edge nodes, such as wireless BSs, routers, IoT gateways, and APs. The bottom edge nodes are responsible for receiving data from end devices of the perception layer and delivering control flows back to the devices via various wireless interfaces [81]. The upper edge servers perform the computation tasks using the received data. The computation tasks can be offloaded to a higher-level server with more powerful computation capabilities and memory if the complexity of the task exceeds the ability of the current server. The control flows of these servers will also be returned to the BSs or APs and finally passed to the end devices. Other functions of edge servers include authentication, authorization, offloading and storage of the data passing between networks. This kind of edge computing can lower latency, provide continuous services without the need for a stable Internet and further protect data security and privacy by avoiding uploading data to the cloud. This is of practical value to AIoT applications, such as agriculture, shipping, and smart grids, without a stable Internet. Compared with traditional cloud computing, the resources of the edge layer are inadequate. However, DNN models are extremely suitable for deployment on edge nodes since the layers of DNN models can be partitioned into several parts to be executed on different nodes [3]. Several techniques have been explored to make model inference work on edge nodes, as is further discussed in Section V-B. In addition, model training can work on edge nodes by using federated learning (FL) or transfer learning (TL), as reviewed in Section VI-B.

3) Cloud Layer: The cloud layer can coordinate the end layer and edge layer, and is responsible for mining potential value from massive data and training AI models. The cloud layer empowers AIoT applications to use virtual resources through the Internet, which provides flexible and scalable

resources on demand, e.g., computation and storage resources. The cloud layer enables various intelligent services, such as the IoV, smart homes, smart grids, smart agriculture, smart industry, and smart healthcare. Typically, the data from end devices are finally transmitted to the cloud via the Internet for further processing and storage, and the AI model will be better trained with higher accuracy. Moreover, deployment of AI models on cloud platforms empowers unintelligent devices with smart control and decision making.

C. Example of Artificial Intelligence of Things Applications

In this section, the HydraMini, an affordable experimental research platform for AD, is taken as an example to describe how AI models can be applied to the real world in an edge-cloud cooperation mood, as shown in Fig. 5. The following describes the corresponding system design, illustrates model training and deployment as well as inference and provides a case study using an end-to-end AI model.

1) System Design of the Autonomous Driving Vehicle: The design and implementation of an AD system consist of a cloud center and the HydraMini edge computing platform, which is designed by the Center for Automotive Research (CAR) at Wayne State University [82]. The cloud layer is responsible for model training, model compression and model compilation, while the edge platform executes the AD tasks. The HydraMini is equipped with a Xilinx PYNQ-Z2 board, where an Advanced reduced instruction set computer (RISC) Machine (ARM) core ARM Cortex-A7 and an Artix-7 FPGA in the same chip are integrated. All the resources are managed by the board, the operation system of which is based on Ubuntu 18.04. Thus, multiple ML libraries (PyTorch, TensorFlow, Open Source Computer Vision Library (OpenCV), etc.) can be easily installed. A producer–consumer model [83] is implemented to make the control process more efficient and easier to extend; in this process, the sensors and decision-making models play the roles of the producers and consumers, respectively. The producers add the data to the specified queue in the memory, while the consumers handle the data. More producers or consumers and new kinds of data can be easily added by reading or writing the related queue in the memory; consequently, various kinds of applications based on different sensors can then be deployed. The consumers usually serve as the controllers who have a shared clock to check whether the vehicle receives the outdated commands. Moreover, a token is used to indicate who is in control at this moment. Users can build strategies for transforming properties. Basic control application programming interfaces (APIs) are provided to set the rotation speed of the motor and the angle of deflection of the servo motor. Additionally, movement of the vehicle can be controlled by using a keyboard, where the OpenCV library is utilized to read the keyboard signals and then call the APIs.

2) Model Training, Deployment and Inference: Movement of the AD vehicle can be controlled by the results of AI inference or the keyboard signals. To enable AI inference to work on HydraMini, the corresponding model has to be pretrained on remote cloud because the resource intensive task will easily exhaust the mobile device owing to a large amount of computation. The training data can be saved from the multiple sensors by controlling the vehicle using the keyboard, and the obtained data serves as input of AI models while the keyboard signals as output labels. Alternatively, the training data can be also downloaded from the Internet. After training, to deploy the pretrained TensorFlow (an end-to-end AI framework) model on HydraMini, the following process are inevitable. Concretely, with the assistance of the DL processor unit (DPU) accelerator in the FPGA, model inference will be accelerated. To make the DPU work more effectively, model compression and model compilation are necessary. The deep compression tool (DECENT) [84] uses coarse-grained pruning, quantization and weight sharing to make model inference run fluently, thereby striking a good balance between latency and accuracy. Then, the DNN compiler (DNNC) [84] is employed to map the neural network algorithm to the Xilinx DPU instructions, aimed at achieving maximum utilization of DPU resources by balancing the computing workload and memory access. Finally, AI inference can run on HydraMini, where AI inference is packed as a consumer thread, obtains data from the data queue, and then is sent to the AI network. The model will directly generate the control commands or send information to the controller thread for decision making.

3) Case Study Using an End-to-End Model: The platform of HydraMini can support AD using end-to-end AI models. End-to-end AI models are simple to execute because the models can directly output commands. As the model structure is shown in Fig. 5, the model consists of CNN and dense layers and maps the camera input to the control output. CNN layers are responsible for extracting features from image inputs followed by fully connected layers, which can be applicable to the field of AD to extract the command information. ReLU serves as the activation function. A softmax layer or a dense layer works as output layer for classification or regression. It is easy to conduct optimizations on the model if the model is far from satisfactory. Fig. 5 presents the whole process. First, users are empowered to use the keyboard to control the vehicle as will and save the data from the sensors as training data. Second, the preprocessing data is transmitted to input of the AI model and the keyboard signals is used as output. To further improve its accuracy, the model is trained in the cloud by using TensorFlow until a perfect model is obtained. Third, the pretrained model is compressed and compilated, and then the copy produces files to the vehicle. Finally, movement of the vehicle is controlled by the output results of AI inference.

IV. Applications of Artificial Intelligence of Things

To date, the utilization of AIoT applications has reached visionary scales, continuing to have profound impact on both the quality of daily life and economic growth. In this section, several application scenarios, including smart homes, smart healthcare, smart agriculture, smart industry, smart agriculture, smart grids and smart environment, are presented to demonstrate how edge computing aided AIoT systems will make

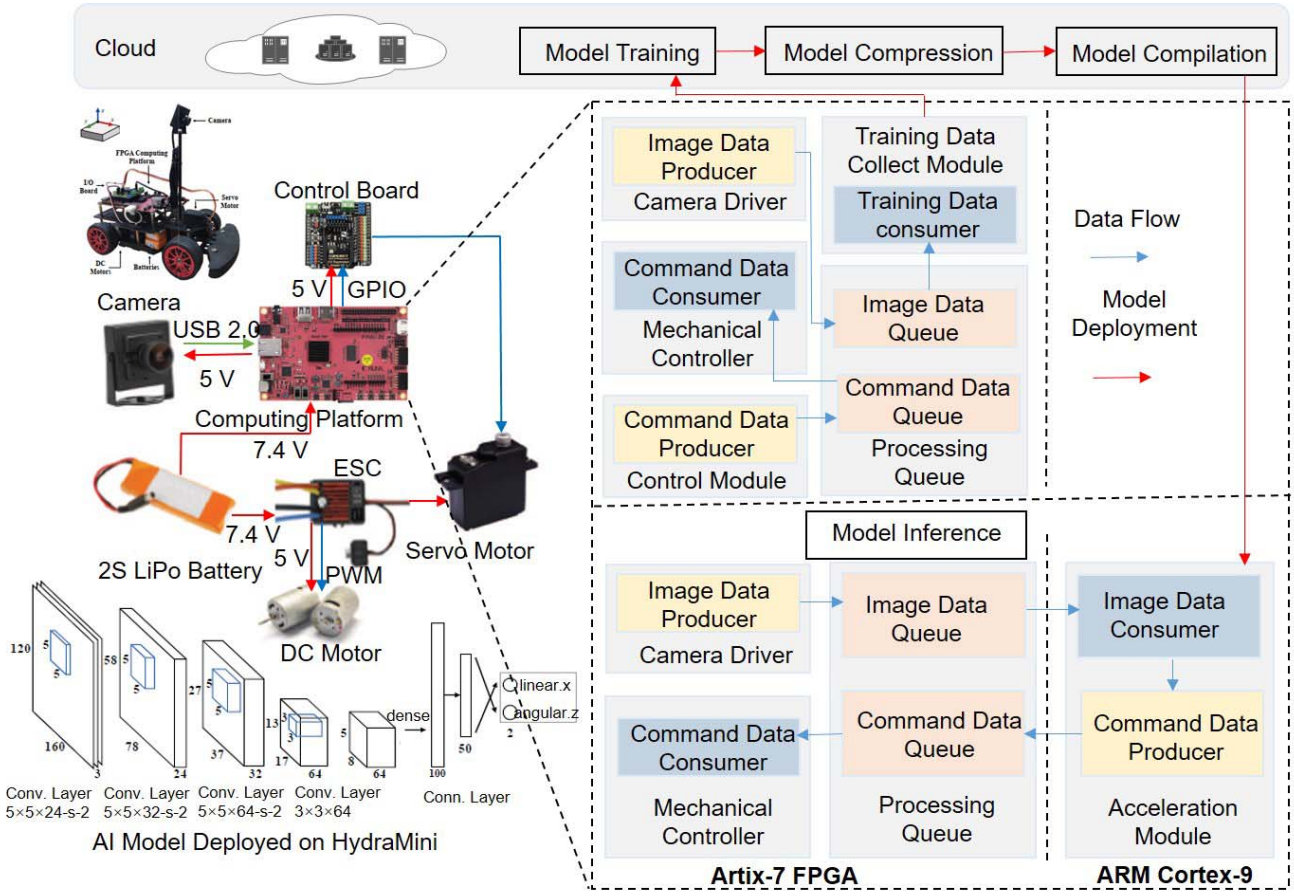

Fig. 5. AD using an end-to-end model [82].

real-world more efficient, smarter and safer. The forementioned application scenarios are shown in Fig. 6. Additionally, the main goals of each field are listed in Table II.

A. Internet of Vehicles

The IoV enabled by AIoT aims to enhance road safety, strengthen efficiency, decrease crash risks and lower traffic congestion in transportation systems. Currently, the IoV covers three major categories: 1) AD; 2) monitoring systems for safe driving; and 3) cooperative vehicle infrastructure systems (CVISs).

1) Autonomous Driving: AD achieves the intellectualization of vehicles, which can sense the environment through on-board sensors and make smart decisions using AI algorithms to control movement of the vehicle by automated means without human labor. An AD car produces 4000 TB of data per day, which is impossible to upload to the cloud [123]. In addition, AD cars need to make real-time decisions. A transmission lag will result in serious consequences. Edge computing is an ideal option that directly moves computing tasks to the edge of the network (e.g., the vehicles themselves). HydraOne [85] and HydraMini [82] are typical examples of AD vehicles, where vehicles are equipped with embedded computing platforms to support AI (e.g., CNN) inference and traditional computer vision analysis and can make real-time decisions (e.g., to shift, brake, throttle, and steer). Moreover, advanced driver assistance systems (ADASs) have been developed to assist drivers. EdgeDrive, an edge cloud-based system, is designed to provide real-time ADAS applications, such as smart navigation, to the driver while driving [86].

2) Monitoring Systems for Safe Driving: Driving requires the cooperation of all the senses acting together, and a slight slip may cause a major accident. Drowsiness or fatigue is the main factor that leads to serious traffic accidents, which deserves more attention. A somnolence detection system is fabricated on a Raspberry Pi 3 using a DL algorithm to make a timely alert when drowsy driving is detected [87]. The proposed system analyzes images captured by a camera equipped with infrared lights using a SqueezeNet DNN to recognize patterns of the driver's facial features (eye closure, nodding/head tilting, and yawning) in both daytime and nighttime.

3) Cooperative Vehicle Infrastructure Systems: CVISs acquire a comprehensive picture of dynamic, real-time road information and share the information by connecting vehicles to vehicles, pedestrians and road infrastructures through the Internet. The edge computing platform includes many distributed infrastructures, such as vehicles themselves, BSs nearby, and roadside units (RSUs). Directly processing V2X data on nearby devices can reduce the transmission delay, and the device will broadcast real-time road condition messages or

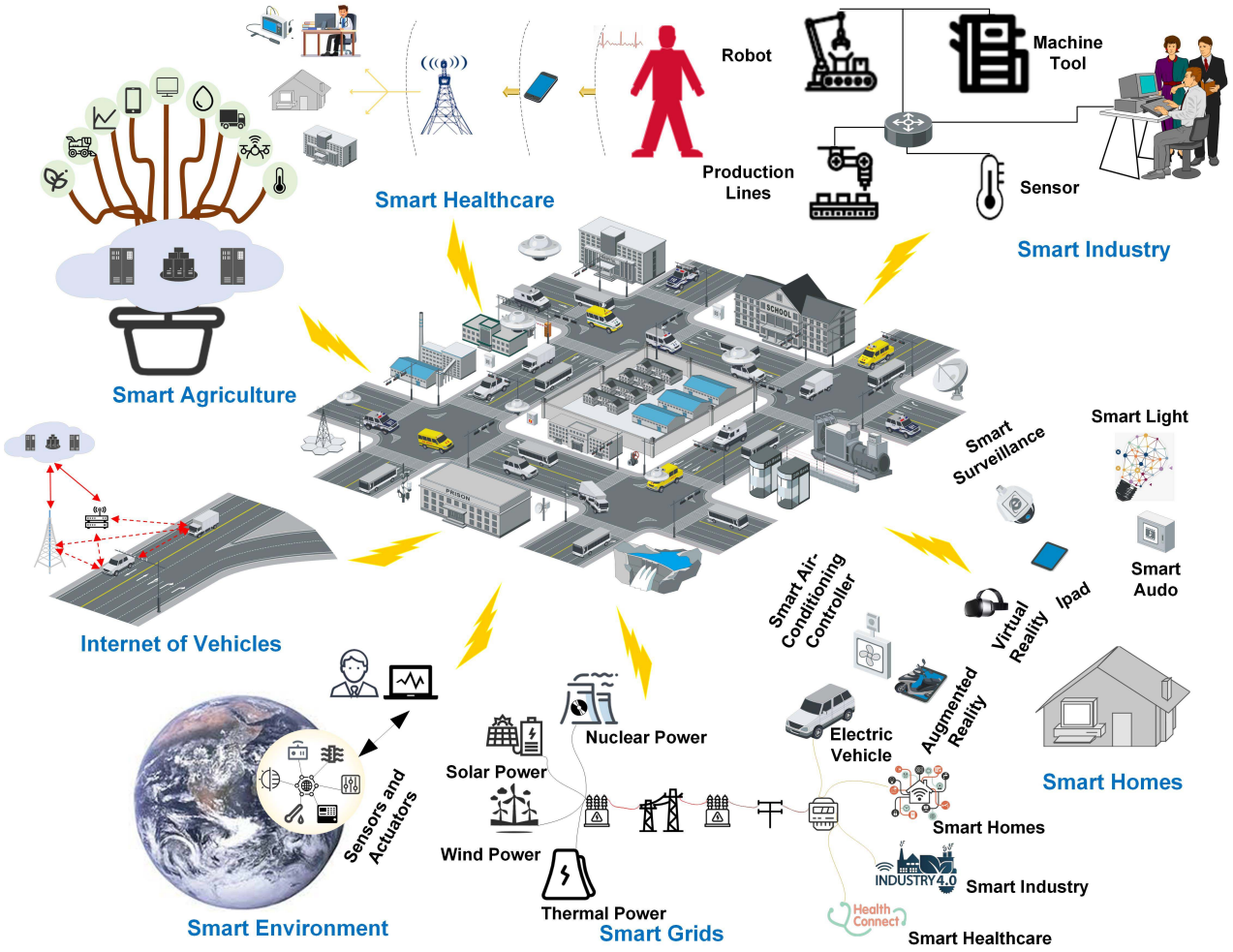

Fig. 6. Examples of an AIoT-driven smart world, including smart homes, smart healthcare, the IoV, smart cities, smart agriculture, smart grids, and smart environment.

traffic accident information to the relevant vehicles and pedestrians near a geographical location. The users can change the route in advance, thereby alleviating road traffic congestion and ensuring safer road traffic. Moreover, aided by device-to-device (D2D) communication and the advantage of a latency less than 1 ms, 5G is suitable for providing direct connection services. For instance, a great deal of road traffic deaths and disabilities are caused by untimely treatment or secondary accidents. To this end, an automatic car accident detection system based on a you only look once (YOLO) DL model for car accident detection (YOLO-CA) is developed in [89]. To improve accuracy of accident detection, the proposed system employs a data set named car accident detection for CVIS (CAD-CVIS) that mainly consists of crash videos from public video sources and the captured images from roadside intelligent devices in CVIS. The RSUs will report traffic accident casualties to rescue agencies and nearby vehicles using 5G networks, which can significantly shorten the rescue time response and improve rescue efficiency.

B. Smart Healthcare

The era of edge-computing-enabled AIoT opens a new line of research in the field of medical and healthcare systems, which has already provided a myriad of applications, including health monitoring systems, disease diagnosis systems, and auxiliary therapy systems.

1) Health Monitoring Systems: The AIoT provides a great opportunity to develop smart health monitoring systems to track patients' health status with respect to psychological and physiological conditions. Smart health monitoring systems employ a wide variety of wearable devices (advanced sensors, smart watches, smart clothes, smartphones, etc.) to collect information (heart rate, blood pressure, blood sugar level, amount of sleep, etc.). These devices can diagnose conditions and sounding a warning [100]. Edge computing is characterized by low latency, real-time responsiveness and privacy security, which is an ideal option for smart healthcare systems. In [101], smart socks are applied to detect falling and symptoms of Parkinson's disease; the socks, equipped with a textile-based triboelectric nanogenerator, are used to capture personalized triboelectric output signals and provide power for data transmission, while nearby smartphones perform real-time comprehensive gait analysis using AI.

2) Disease Diagnosis Systems: Emerging applications of AIoT have been explored in disease diagnosis systems to help doctors diagnose diseases more accurately and reduce

TABLE II
AIoT APPLICATIONS APPLIED IN DIFFERENT AREAS

Domains	Ref.	Main Goals
IoV	[82], [85]	• Intellectualizes vehicles, thereby enabling vehicles to control movement themselves by AI technologies
	[86]	• Advanced driver assistance systems aimed at providing intelligent services, such as smart navigation
	[87], [88]	• Safe driving monitoring systems that prevent drivers from becoming distracted
	[89]	• Cooperative vehicle infrastructure systems that ensure safer road traffic and alleviate road traffic congestion
Smart Homes	[90], [91]	• Remote monitoring systems
	[92]	• Intelligent control systems that increases convenience and improves life quality
	[93]	• Smart guard systems
Smart Agriculture	[94], [95]	• Timely detects diseases to avoid the wide spread of epidemics
	[96], [97]	• Environment monitoring to help farmers make scientific decisions and ensure profitability, sustainability, and protection of the environment
	[98], [99]	• Intruder detection systems that help prevent wild animals from destroying crops
Smart Healthcare	[100], [101]	• Health monitoring systems that precisely monitor patients' physical and mental condition
	[102]–[104]	• Disease diagnosis systems that help doctors to accurately detect diseases
	[105], [106]	• Robot-assisted surgery
Smart Grids	[107]	• Detects faults for real-time alerts and takes immediate action
	[108], [109]	• Intelligent attack detection systems
	[110]	• Energy management systems that enhance the efficiency, reliability, and economy of smart grids
	[111], [112]	• Economy-driven vehicle charging systems that automatically determine a charging policy
Smart Environment	[113], [114]	• Real-time environmental monitoring systems
	[115], [116]	• Disaster detection systems that send early warnings
	[117], [118]	• Waste management systems
Smart Industry	[119]	• Provides high-accuracy intelligent systems, aimed at increasing efficiency and productivity in assembly/product lines and reducing maintenance expenses and operational costs
	[120]	• Inspection systems aimed at effectively determining potential defects
	[121], [122]	• Industrial data analysis aimed at making accurate predictions

the burden on doctors. Currently, most disease diagnosis systems are designed under the cloud framework, where the data are transmitted to the cloud via 4G networks and Wi-Fi for further analysis using AI technologies. However, this inevitably leads to a decrease in system performance, such as increased latency and privacy leakage. HealthFog is proposed to solve this problem, aimed at real-time automatic heart disease analysis by integrating DL algorithms in edge devices [102]. Additionally, due to the emergence of new infectious viruses and diseases, early detection and remote monitoring are necessary to control infection. A fog-assisted cloud-based Chikungunya virus diagnosis system has been used to prevent virus outbreaks [103]. On their smartphones, users receive alerts from the fog layer about infections and can avoid proximity to infected regions. Furthermore, an edge-and-cloud-based DL platform using a 5G network is proposed to detect Coronavirus Disease 2019 (COVID-19) in real time by using chest X-ray or computed tomography (CT) scan images and to monitor social distancing, mask wearing, and body temperature [104].

3) Auxiliary Therapy Systems: Robot-assisted surgery (RAS), a kind of auxiliary therapy system, has been accepted by patients and surgeons; robotic surgical tools are controlled by the surgeon's real-time hand movements and operate in small-scale movements. It is easier to perform complex motion tasks, and there is a faster recovery rate with less pain. However, RAS is confronted with the challenges of narrow vision and operating space and holes in the organs and tissues during the operation. To address these issues, surgical tool detection based on anchor-free CNN architectures on a TITAN GPU is proposed for real-time automated surgical video analysis to assist surgeons with automatic report generation and optimized scheduling [105]. Moreover, in [106], an NVIDIAGTX 1080Ti GPU is used to build an iris tracker for the ophthalmic robotic system by using a CNN model, which helps surgeons with reference locations for incisions and protects patients.

C. Smart Industry

The fourth industrial revolution, also known as Industry 4.0, has created new opportunities for product robotization and automation, which emphasize intelligent manufacturing techniques. The edge-computing-aided AIoT caters to the requirements of smart manufacturing, where edge computing enables low-latency, secure manufacturing while AI provides more intelligent local analysis and prediction. Smart industry focuses mainly on production automation and smart data analysis.

1) Smart Manufacturing: Smart manufacturing aims to provide high-quality and low-cost production lines through the intelligent decision of AI and low latency processing of edge computing. Edge-powered AIoT in smart manufacturing has been applied in a variety of applications, such as product assembly lines and fault diagnosis. An intelligent robot factory is designed and developed where routers and gateways serves as edge computing node to relieve network transmission time and achieve increased production during manufacturing and production [119]. Inspired by edge computing, an inspection system based on a CNN model coupled with an early exit technique is deployed on a fog server to effectively discover potential defects and measure their degrees, thus showing improved inspection accuracy with low latency [120].

2) Smart Industry Analysis: Edge-empowered AIoT is also suitable for industrial data analysis. An assembly prediction system aided by edge computing is formulated to address the problem of high-dimensional and imbalanced data, where the

random forest algorithm is used to reduce dimensionality and extract characteristics, while a synthetic minority oversampling technique–adaptive boosting (SMOTE-Adaboost) method with jointly optimized hyperparameters is applied for imbalanced data classification [121]. The use of edge computing can lower the latency of data transmission and enhance the flexibility of application deployment and the efficiency of quality prediction. Faults and errors often bring great loss for factories. To reduce loss, a predictive maintenance framework is triggered for conveyor motors to detect the early stage of faults and errors in machinery [122].

D. Smart Homes

The rapid development of AIoT has encouraged many attractive computationally intensive applications that provide intelligent sensing and convenient control services in smart home scenarios. For home data protection, edge computing has emerged as an excellent option to execute local computation and processing, especially for some computation-intensive AI-based applications.

Home monitoring systems are developed to reduce the concerns of family members by identifying and analyzing the behavior of specific family members or house conditions. Usually, it is difficult for children to finish complex homework alone. An intelligent home environment employs a smart chair and desk to collect the child's real-time behavior information and a robotic assistant to interact with the child as a therapist would do [90]. The implementation of such a system can provide not only education for children without pathologies but also therapeutic interventions for children with various pathologies. Wi-Fi signals exist in almost every corner of a home; thus, Wi-Fi-based activity recognition has received constant attention from researchers because of its ubiquity, low cost and device-free experience advantages. A Wi-Fi-based moving human activity recognition system is explored using a combined CNN and LSTM model [91]. Such a method can be applied to the intelligent control of home devices, such as lights and air conditioners [92]. A smart floor monitoring system based on self-powered triboelectric floor mats can be used in position sensing, activity monitoring and individual recognition by using a DL-based analysis method based on instant sensory data [93]. Wi-Fi-based activity recognition also demonstrates great potential in theft detection [124].

E. Smart Agriculture

Smart agriculture aims to improve crop yield and quality, reduce labor costs and protect the environment from the excessive use of pesticides and fertilizers by using modern technologies. The explosive employment of sensors and automated equipment will generate an abundance of data, thereby taxing the Internet and cloud center. Edge-computing-aided AIoT applications enable data to be processed locally or on nearby edge servers to seek a timely response. Generally, smart agriculture concentrates on crop production, agriculture environment monitoring and agriculture security.

1) Crop Production Analysis: Crop production analysis can promptly detect diseases to stop the wide spread of epidemics by monitoring their growth. Deep Leaf, a quantized CNN model for timely detection of early symptoms of coffee leaves, is developed using the X-CUBE-AI tool on an edge device, such as the STM32 family [94]. A real-time apple detection using a DL method has been designed on a Raspberry Pi 3 B+ with improved energy consumption, and inference time [95]. The embedded device is equipped with various kinds of accelerators to carry out DL inference, while a dedicated workstation is equipped with an NVIDIA RTX 2080Ti GPU to train DL models, where the YOLOv3 tiny architecture is exploited to accurately recognize, count, and measure the size of apples. This kind of method can be applied on the AD cars to monitor the crop growth on a large scale.

2) Agriculture Environment Monitoring: Smart agriculture can accurately measure and predict the crop growth environment that influences crop productivity, thereby helping farmers to make scientific decisions. To predict the temperature of the crop growth environment in a real time, an edge-computing-aided smart agriculture system is designed by measuring the change in the CPU temperature of single board computers. Here, the edge server is used to collect the CPU temperature, where outdoor temperature is predicted using a combination of single spectrum analysis and linear regression [96]. Furthermore, productive soils significantly affect crop yield. The concentrations of nutrients in soil are measured by using colorimetry, where the data sensed by the nitrogen-phosphorus-potassium (NPK) sensor is directly processed by a fuzzy rule-based system on a Raspberry Pi 3 [97]. Such an approach can warn farmers of the deficiency of N, P and K available in soil, help to optimize the soil fertility and avoid water contamination by runoff and leaching. Additionally, more AI algorithms are also applied to manage the crop growth environment.

3) Agriculture Security: Agriculture is the foundation of a country's economic development and provides food and other raw materials. Thus, protecting agriculture from malicious damage is crucial. To handle animal-human cohabitation, a CNN-based early warning system is proposed with the support of the IoT and edge computing, where the sensed or processed data can be further transferred to BSs via a fiber-wireless network to alert humans of possible animals crossings [98]. In addition to the function of selecting suitable crops according to the soil conditions, the edge computing-assisted agriculture system can also keep trespassing wild animals away from the agricultural area, where a CNN model is used to detect animals by using the captured images on a Raspberry Pi 3, and a buzzer is utilized to deter intruders by irritating them [99].

F. Smart Grids

Grid operators have deployed IoT sensors to monitor power grid devices in real time. Recently, there has been a rush to integrate AI with electrical grids to provide a more stable, cost-saving and secure smart grid. The Linux Foundation's LF Energy releases a scalable and technology-agnostic industrial IoT platform, grid exchange fabric (GXF), enabling grid operators to securely gather data and monitor, control and manage

smart devices on the grid [125]. Independent system operator (ISO) New England (ISO-NE) has successfully used cloud computing technology for large-scale power system simulations, which may be capable of dealing with the increasing challenges [126]. The smart grid application, distribution management systems, can significantly benefit from the usage of cloud [127]. Moreover, attention is drawn to edge-computing-based solutions for smart grids. An edge-assisted AIoT smart grid architecture is fully explored to support the connection and management of substantial terminals, data privacy protection, and real-time data analysis and prediction [128]. For automated fault detection, an intelligent damage classification and estimation system is proposed to automatically localize and predict damage to power distribution poles; the system processes images captured by unmanned aerial vehicles (UAVs) by using a CNN model [107]. A more transactive energy system is desired to promote efficiency, save cost and time, and ensure delivery; the system is also desired for the environmental advantages gained from the increasing use of intermittent renewables [129]. For example, renewable energy management is designed using an LSTM-based model to predict solar intensity, guarantee a power balance and improve the reliability and efficiency of smart grids [110]. In addition, pricing is an important factor that significantly influences users' behaviors, such as in the case of economy-driven electric vehicle charging. Considering the electricity price and the battery energy load, TL [111] and DRL [112] methods are utilized to automatically determine vehicle charging policies.

The security of smart grids is also noteworthy. A DL-based mechanism that a deep belief network (DBN) and restricted Boltzmann machine (RBM) is exploited to detect potential false data injection attacks in real time [108]. Moreover, online attack/anomaly detection issue is formulated as a partially observable MDP to accurately detect cyberattacks by using an RL-based detection algorithm [109]. Although a cloud-centric energy theft detection scheme is used to detect abnormal behavior in smart grids, the scheme still requires privacy preservation techniques during data transmission, which must be explored in edge computing scenarios [124].

G. Smart Environment

The goal of smart environment is to provide humans with a safer and more comfortable life. Environmental monitoring systems can send early warnings to humans by making accurate predictions. To monitor air quality, an AI-based model is designed using on-chip sensing systems to predict the concentration of Particulate Matter with a diameter of 2.5 μm or less (PM 2.5) [113]. A seawater quality assessment integrating principal component analysis (PCA) and a relevance vector machine (RVM) are developed in an edge computing environment to predict the potential values of dissolved oxygen and pH [114]. This type of method can be used for city water prediction. A smart environment can also be applicable to accidental disaster detection. A CNN-based model is proposed on an NVidia GeForce GTX 1060 to detect fire and smoke and thus provide warnings in advance [115]. UAVs are also employed to increase the area of detection and are suitable for forest fire detection [116]. Many studies have explored cloud-based geological hazard detection systems designed to prevent the effects of the disasters [55]. And more work should focus on edge-computing-assisted detection systems designed to provide warnings more quickly.

Smart environment can help manage a city. For instance, much effort has been made regarding waste management systems to mitigate the side effects of waste materials. A scheme fusing multiple deep models is exploited to classify waste [117]. To reduce latency and enhance availability, the research presented in [118] implements a DL-based waste detection model on a Raspberry Pi 3 Model B+. WasNet adopts a lightweight neural network to accurately sort waste, and then transplanted to the hardware platform and transformed into smart trash [130].

V. Enabling Technologies for Artificial Intelligence Inference in Artificial Intelligence of Things

Most AI models, especially DNN models, are designed to be much deeper, which have a larger data set to promote their accuracy. Inference in the cloud will inadvertently incur additional queuing and propagation delays from the network, which is fatal for time-critical applications. These AI models, however, are too large and computationally expensive to be directly deployed on resource-constrained end devices. To overcome this challenge, one possible approach is to simplify the models with a dramatic decrease in computation. The other effective approach is to outsource complex inference tasks to edge nodes with more resources. In this regard, the methods used to optimize inference on device and coinference in the edge, and privacy-preserving techniques, are surveyed.

A. On-Device Inference

Generally, two practical methods can be used to reduce computational costs of AI models. One is to directly design compact and efficient neural network models with a reduced number of parameters, such as SqueezeNet [131], Xception [132], and ShuffleNet [133]. The other method is typically called model compression, which achieves a smaller memory footprint and improved operation for end devices by compressing pretrained networks.

1) Designing Compact Networks: Researchers usually focus on constructing a compact DNN model to reduce the number of parameters while maintaining impressive accuracy. Many measures have been taken to achieve this purpose, and the well-accepted DNN models widely used on resource-constrained devices are presented, including Xception [132], MobileNets [134], YOLO [135], and SqueezeNet [131]. Xception and MobileNet use depthwise separable convolutions, originally based on the idea of factorized convolution instead of the standard convolution operation, to obtain a reduced number of computations and parameters for neural network models while maintaining high performance [134]. SqueezeNet utilizes special 1×1 convolution filters to replace the original 3×3 convolution filters and downsamples the

number of input channels [131]. To overcome the disadvantage of MobileNet's group convolutions, ShuffleNet adopts a novel channel shuffle operation to help information flow across feature channels [133]. YOLO is a single-shot detector that integrates target area prediction and category prediction into a single neural network model to achieve real-time, fast target detection and recognition with high accuracy, thereby proving that integrating target area and category predictions is much faster than completing these steps sequentially [135].

2) Model Compression: Model compression makes inference reduce the complexity and resource requirements of models with a slight accuracy loss. Below, efforts toward model compression techniques are briefly discussed.

a) Parameter pruning and sharing: Parameter pruning and sharing can decrease the number of redundant parameters and address the issue of overfitting. Model pruning methods are roughly divided into structural pruning and nonstructural pruning. Nonstructural pruning is generally a connection-level, fine-grained pruning method with relatively high accuracy; however, it depends on a specific algorithm library or hardware platform, such as deep compression [136] or sparse-winograd [137]. Structural pruning is a filter-level or layer-level coarse-grained pruning method with relatively low accuracy, and the structural pruning strategy is more effective than nonstructural pruning and does not depend on a specific algorithm library or hardware platform. This pruning strategy can run directly on DL frameworks, such as ThiNet [138] and network slimming [139]. A channel pruning method is designed that convolutional layer is pruned by least absolute shrinkage and selection operator (LASSO) regression-based channel selection and least-squares reconstruction [140]. A pruned VGG-16 can reduce the number of weights by 5× with a slight increase in the error, while this method suffers only 1.4% and 1.0% accuracy loss under a 2× speedup for the residual neural network (ResNet) and Xception networks.

b) Quantization: Quantization uses a more compact format by adopting low-bit width numbers instead of 32-bit floating-point numbers to represent each weight, thereby reducing the computational intensity and the memory footprint and further increasing the energy efficiency. It is sufficient to use fixed-point networks, compared to their 32-bit full-precision floating-point counterparts, with a negligible degradation in performance [141]. Some quantization methods for DNN models are confined to limited types [141]–[143] and bound to a specific hardware and DNN framework [142], [143]. To address this issue, *libnumber*, a portable and auto-tuning framework, is proposed to optimize number representation for each layer of DNNs; the abstract data type (ADT) of the portable API allows users to declare the data (e.g., inputs, weights, or both) to be quantized in a layer as number type [144]. Then the auto-tuner of *libnumber* will use a compact representation for the number to minimize the user-supplied objective function with a certain accuracy remaining. In this way, the concern of developing an effective DNN model is totally separated from low-level optimization of the number representation.

c) Knowledge distillation: Knowledge distillation fabricates a compact DNN model that migrates the behavior from a powerful and complex DNN model. By training the smaller DNN model by using the output predictions generated by the complicated model, the smaller DNN model should approach or exceed the function trained by the larger DNNs as well as possible, as illustrated in Fig. 7. The framework of knowledge distillation is first introduced in Hinton's work, where a "softmax" output layer is employed and a temperature T is used in the softmax layer to produce a softer probability distribution over the classes and improve the distillation performance [145]. A hint-based training method, an extension of knowledge distillation, is carried out by using the intermediate representations learned by the teacher as hints, thus making the training of the student deeper and thinner [146].

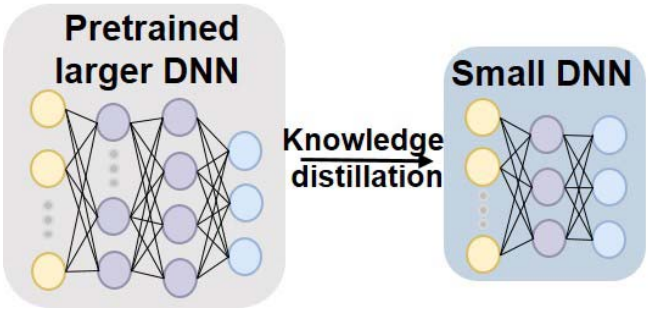

Fig. 7. Knowledge distillation method for DNN on-device inference.

A stand-alone compression technique may not meet the requirements of computation- and memory-intensive end devices. Thus, some efforts have focused on reasonable combinations of these model compression techniques. Deep compression is employed by using a combination of weight pruning and trained quantization to reduce the storage requirements of a neural network without loss of accuracy [147]. Additionally, an RL-based optimizer that effectively chooses a good combination of various model compression techniques achieves a balance between the application requirements and mobile resource constraints [148].

B. Coinference at the Edge

The forementioned works concentrate mainly on designing compact neural networks and model compression techniques, which can make AI model inference work directly on end device without an apparent loss of accuracy. However, deploying large-size AI models, which require high computation, power and memory capacity from the end infrastructures, remains a challenge. Therefore, it is a good option to segment DNN models into multiple partitions and offload each part to heterogeneous local end devices, more powerful distributed edge servers or remote cloud servers. In what follows, the enabling technologies for such coinference between the end layer and edge layer are summarized.

1) Offloading: Computation offloading is a widely used distributed computing paradigm in fast inference, where an end device can migrate part of its computation to an edge node, the cloud, or both over a heterogeneous network. By this means, offloading can employ remote servers to increase the

computation speed and save energy. However, a compromise between advantages in remote execution and sacrifices in data transmission should be reached. A great deal of optimization-based offloading approaches, such as DeepDecision [149] and mobile-cloud DNNs [150], have been taken into consideration, which have strict constraints on network latency, bandwidth, energy consumption, accuracy and the input size of the DNN models. Moreover, the decision of whether to offload or not also depends on the input data size and hardware capabilities.

Offloading technology has been widely discussed in the context of networking [151] and edge computing [152]. An irreversible trend has been to explore such technology in AIoT. Glimpse, a real-time object recognition system on mobile devices, ships the execution of algorithms for object recognition to server machines to reduce latency [153]. In addition, offloading can also take place in the same tier. The femto-cloud system configures a cluster of co-located mobile devices into an orchestrated cloud service [154]. Such an offloading approach enables resource-constrained IoT devices to outsource the computation intensive tasks to other mobile devices nearby rather than to a MEC server. It dedicates advantages includes better scalable computational capacity and not relying too much on edge servers. However, the dynamicity, and instability of mobile devices make security unavoidable challenges. It is still a great option to offload computation tasks to a more stable server. GigaSight, a framework for crowd-sourced video analysis, removes privacy-sensitive information in a user-specific vector machine on the cloudlet and then runs part of the computation in the cloud [155].

2) DNN Model Partitioning: Offloading techniques shift model inference to edge servers or cloud servers, which is highly dependent on unpredictable server availability and network conditions. The idea of offloading can be extended to model partitioning, which takes advantage of the unique structure of DNNs. In this way, the layers of DNNs can be divided into several parts, where some layers are directly executed on end device and some layers are offloaded to edge server or the cloud for remote computation. This approach can provide improved performance with latency reduction and energy efficiency enhancement. Generally, the DNN model can be decoupled into multiple partitions that are allocated to 1) distributed mobile devices [156]; 2) edge nodes [157]; or 3) the cloud [3], as illustrated in Fig. 8.

Partitioning DNN models horizontally by layers is the most common method, where some layers are executed on devices and some layers are implemented on edge nodes or the cloud. This can decrease the power consumption of the IoT device and the latency by using the computation cycles of other end devices, while there exists a delay in exchanging the intermediate results at the DNN partition point, which can still generate overall net benefits. Care must be taken regarding where to partition the DNN model. Neurosurgeon is a system that applies to DNNs and intelligently partitions DNN models at the granularity of neural network layers to obtain the best latency and end device energy consumption [3]. An edge computing system for object tracking is conducted by [158] where the best partition point between end device and edge server is automatically selected to minimize the power consumption

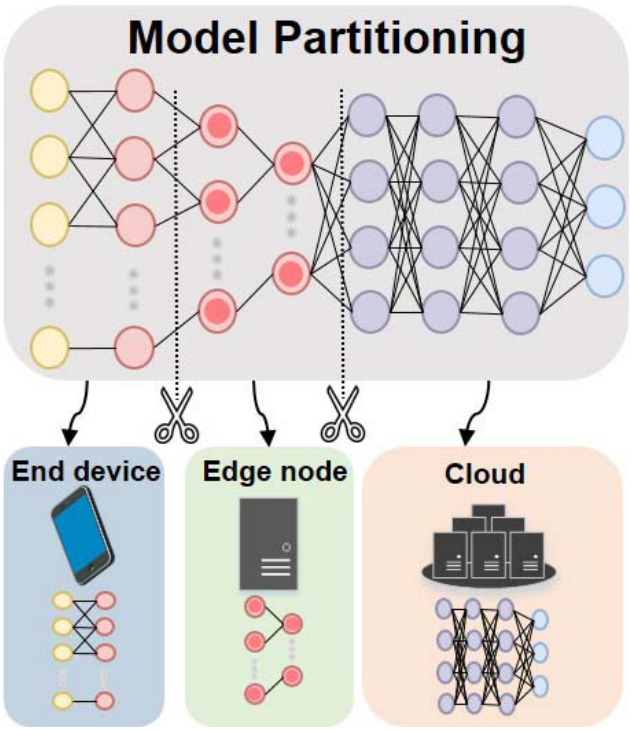

Fig. 8. Model partitioning for DNN inference at the edge.

of IoT device and the latency of dynamic network bandwidth. CNN models can be vertically partitioned. DeepThings, a lightweight system for distributed implementation of CNN inference on a resource-limited device, employs a fused tile partition method and divides the CNN models into separate distributable subtasks to decrease the memory footprint [157].

3) Model Early Exit: A DNN model with additional layers can generally achieve higher accuracy; however, the model requires increased computation and energy resources in feed-forward inference. Therefore, it is difficult to execute such a complicated DNN model on a resource-constrained end device. Various approaches, such as offloading and model partitioning, have been adopted to make inference work on edge servers, thereby reaching a balance between accuracy and processing latency. The idea of accelerating model inference can be further promoted by the emerging model early exit method, which leverages additional side branch layers to obtain the classification result [159]. The inference process can be completed in advance via the early classifiers with high confidence. It is also possible for some complicated tasks to use more DNN layers to complete the classification procedure.

As seen in Fig. 9, by using the technology of model early exit, a partial DNN model on a device can quickly extract features and, if it is confident, directly produce the inference result [120]. Compared to offloading DNN computation to the cloud, model early exit can decrease the communication delay [120], [160]. Moreover, this method achieves a greater inference accuracy than the forementioned methods (parameter pruning and quantization) for model compression on an end device. Model early exit should not be considered independent of offloading and model partitioning but should be thought of

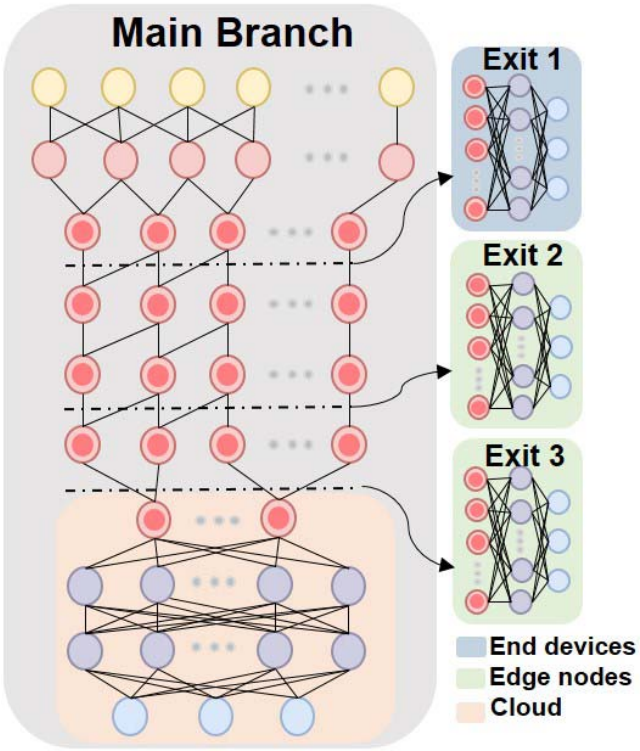

Fig. 9. Model early exit method for DL inference at the edge.

as a cooperation framework between the end, edge and cloud, where the additional classifiers deployed on the cloud further process the features and arrive at the improved result [161]. This method also provides better privacy protection because the extracted features rather than the original data are sent to other devices.

To meet the strict latency requirements of AIoT applications, attempts have been made to perform model inference near the data source. The approaches can be generally categorized into two types, as summarized in Table III. The first type is to directly deploy inference on end device by designing compact networks or compressing existing ones. The compact models target reducing the number of parameters while maintaining impressive accuracy, while model compression techniques tend to remove redundant structures and parameters without significant loss of model performance. The parameter pruning and sharing method has proven to be effective in removing redundant and uncritical parameters. Quantization adopts low-bit width numbers to reduce computational intensity and memory footprint. Knowledge distillation aims to learn a distilled model and train a more compact lightweight network to provide accurate inference. These techniques can be combined to seek improvement in model performance. The second approach is to support a cooperative mode of inference between end devices and edge servers with relatively abundant resources, such as offloading and DNN model portioning. Offloading uses remote servers to accelerate inference at the cost of data transmission. Based on the unique structure of DNNs, model partitioning approach segments the model into multiple parts and allocates them to each device or node for coinference. Model early exit enables inference work to exit early via additional side branch classifiers with high confidence and employs more layers to acquire better results.

C. Private Inference

The AIoT collaborative inference techniques between end devices and servers are reviewed above, which can greatly reduce communication costs and latency. However, this kind of cooperative inference system also faces privacy concerns when the data including sensitive information is transmitted to a nearby edge server or cloud. Additionally, end devices are too energy and resource constrained to execute complex data protection methods. Intuitively, it is worth studying privacy enhancement between end devices and edge servers. Initial efforts have been made to protect data from eavesdroppers. In this section, secure computation through encryption or cryptography and data obfuscation techniques are proposed for the AIoT inference.

1) Secure Computation: Cryptography-based methods can be applied to AIoT privacy inference. Edge servers perform computation by using the data preserved by cryptographic techniques while knowing nothing about the data, and end devices receive the result of inference without knowing the model.

Homomorphic encryption can be used in AIoT applications for secure computation, where DNN computation is deployed on the encrypted client data. The commonly used nonlinear functions in DNN models, however, are incompatible with the operations (addition and multiplication) employed in leveled homomorphic encryption. AboulAtta *et al.* [162] trained a model with a nonlinear function and uses a low-degree polynomial approach in the inference of private time instead, inevitably leading to a loss of accuracy. Thus, a novel min-max normalization strategy is induced that limits function inputs to ranges with a low approximation error. Additionally, secure multiparty computation is applicable for AIoT inference privacy, where the multiple parties involved work together to jointly evaluate a model. The efficient priority-preserving scheme (EPPS) [163] trains secure models, supported by the foundation of threshold Paillier encryption, where the secret key is split across each party and a certain number of users collude with one another to infer the privacy of trusted users without the influence of someone exiting at any step of the process. However, secure multiparty computation often leads to a large communication overhead. More studies have exploited a hybrid of homomorphic encryption and secure multiparty computation for AIoT privacy inference [164]. Gazelle [164] proposes using a combination of packed additively homomorphic encryption and garbles circuit-based multiparty computation techniques, where a homomorphic encryption library offers efficient implementations of basic homomorphic operations and homomorphic linear algebra kernels support the efficient use of the automorphic structure. The hybrid framework obtains a much lower latency and bandwidth than the purely cryptographic primitive.

2) Data Obfuscation: To avoid the heavy computation of cryptographic primitives, data obfuscation can provide a strong

TABLE III
SUMMARY OF AIOT MODEL INFERENCE TECHNOLOGIES

Architecture	Technology	Highlights	Ref.
On-device	Compact Networks	• Adopts a compact module in each layer, thereby reducing the number of parameters and amount of computation	[131]–[135]
	Parameter Pruning and Sharing	• Removes redundant parameters that are insensitive to the performance	[136]–[140]
	Quantization	• Uses low-bit data computation to achieve a higher computing speed and lower power without apparent accuracy loss	[141]–[144]
	Knowledge Distillation	• Employs a smaller student network with distilled knowledge of a larger teacher network	[145], [146]
Cooperation between devices and servers	Offloading	• An adaptive offloading strategy that seeks a tradeoff between energy, accuracy, latency, and input size for different DNN models	[149]–[155]
	DNN Model Partitioning	• DNN layer adaptive partitioning that allows cooperation between the resources of the device, edge and cloud • Latency- and energy-oriented optimization	[3], [157], [158]
	Model Early Exit	• Supports partial DNN model inference based on the accuracy requirements • Accuracy-aware	[120], [159]–[161]

guarantee for sensitive data by adaptively injecting noise into data set while retaining the servers' ability to implement AIoT inference tasks. Noise is considered an additional trainable set of parameter probabilities, which can be eliminated by the repetitive process of end-to-end self-supervised training. By adding noise as a gradient-based learning process to discover the distributions of the noise tensors, SHREDDER [165] reaches an asymmetric balance between accuracy and privacy to minimize the loss of accuracy while maximally reducing the information content of the data unloaded by devices to the servers for inference implementation. Differentially private DL algorithms prove to be an effective method, where random noise is added to the data. Edgesanitizer [166] is proposed to protect private data from sensitive inference, where a DL model is used to decrease the data to a defined size while local differential privacy (LDP) is achieved by injecting additional noise to obfuscate the learned features. To avoid the decrease in model accuracy caused by differential privacy, Gaussian and Laplacian noise is added to the input layer and the intermediate layer output and decryption keys are generated to remove random noise in [167]. A two-edge-server framework is also designed for AIoT privacy-preserving inference, where the DNN layers that require much computation are outsourced, while computation-efficient layers are executed on the device.

VI. ENABLING TECHNOLOGIES FOR DISTRIBUTED ARTIFICIAL INTELLIGENCE TRAINING IN ARTIFICIAL INTELLIGENCE OF THINGS

Conventionally, the training mode of AIoT models relies on a centralized style, which may incur additional costs in data transmission and privacy issues. To effectively address these issues, a decentralized training mode is proposed, where the AI model is divided into several subnetworks and each part is trained directly on end device with local data. The trained model updates can be aggregated at edge nodes in the network or be exchanged through the interconnect end devices in the network. The two kinds of decentralized training modes can be realized without the support of the cloud. In this section, enabling techniques for decentralized AI model training, communication efficiency and security enhancement in AIoT are mainly discussed.

A. Decentralized Artificial Intelligence of Things Model Training Methods

The AI inference techniques mentioned above work smoothly on end devices and edge servers of the AIoT system on the assumption that the AI model has been trained in the cloud center by using the existing data set. Increasingly more data are generated at the edge of the network, which is of great significance to AI model training. With respect to the AIoT, it is essential to develop decentralized AI training methods that can avoid data transmission, further reducing the transition bandwidth and enhancing privacy. In this subsection, enabling techniques for decentralized AIoT model training are introduced.

1) Federated Learning: FL is a collaborative AI setting that is originally aimed at addressing the problem of Android mobile terminal users updating models locally with unreliable and slow network connections [168]. Gradually, FL has come to assist in efficient AI training by using data distributed over a large number of end devices and edge servers while ensuring information security, protecting terminal and user privacy, and adhering to legal requirements during data exchange. Traditionally, in the distributed configuration of FL, the AI model is built without direct access to data, while mobile devices serving as clients carry out local training. Additionally, these mobile devices can be extended to end devices, while edge nodes and cloud servers are equivalently considered as clients. A server coordinates a series of nodes, thereby enabling the clients to take responsibility for various levels of ML model training and share the individual trained models with the server [169]. The server creates a federated model by using the uploaded trained models and returns the optimized model to the clients. In addition to DNNs, other important algorithms, such as random forests and so on, can also be trained using FL. As shown in Fig. 10, in the architecture of FL, a set of edge nodes downloads the global DL model from the aggregation server, train their local DL models based on the downloaded global model with local data and upload the trained model to the server for model averaging. Privacy

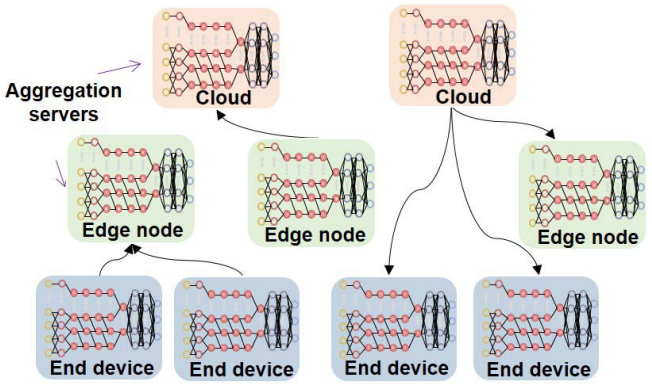

Fig. 10. FL for DL training at the edge.

and security can be enhanced by restricting the local data to be trained solely on end devices or end devices nearby [170]. A decentralized method of FL for CNNs and LSTMs based on iterative model averaging is presented in which the local data are left on mobile devices and the trained models are shared by aggregating locally computed updates [169]. An extensive empirical evaluation on the basis of five different model architectures and four data sets demonstrates that FL is robust to nonindependent and identically distributed (non-IID) data distributions and accelerates the process of training.

2) DNN Splitting: DNN splitting exchanges partially processed data instead of raw data between end devices and edge servers, which is an effective way to protect privacy-sensitive data [21]. The successful deployment of DNN splitting lies in the mechanism by which the DNN model can be split between two successive layers with two partitions executed at various locations with no apparent accuracy loss. However, selecting an appropriate splitting point to meet the requirement of latency remains as a research point. To reduce the computational complexity with accuracy preserved and introduce a bottleneck, it is proposed in [171] to employ network distillation to distill the head portion of the split model. This approach deploys lightweight models on end side and pushes the intensive computation of DNN to the server, minimizing processing load at the mobile device as well as the amount of wirelessly transferred data.

3) Transfer Learning: Knowledge transfer learning, also known as TL, has emerged as a practical DNN training mechanism that enables the convolution kernels to be initialized with the weights learned from the pretrained model and solves the problem of training data drawn from different distributions. TL is closely connected with DNN splitting, the goal of which is to reduce the energy cost of DNN model training on mobile devices and suitable for general-feature image recognition. Extensive studies have explored the impact of TL techniques on DNN training of end devices. Sharma *et al.* [172] provided research on the performance (in terms of both accuracy and convergence speed) of TL and focuses on various student architectures and enabling techniques for transferring knowledge from the teacher to the student. The effect of TL varies with the architecture, transferring techniques, knowledge and skills from both the intermediate layers and last layer of the teacher to a shallower student.

In this section, three distributed AI model training techniques for AIoT are reviewed, the highlights of which are presented in Table IV. FL preserves privacy by leaving raw data on local devices and trains a shared model on the server by uploading the computed updates. Rather than transmitting the raw data, DNN splitting selects a splitting point, thereby enabling distributed DNN models to be trained using the partially processed data. TL applies the general features learned from a DNN pretrained on the basic data to a specific data set or task.

B. Enabling Technologies for AIoT Model Training Updates

Distributed AI training techniques are extremely suitable for scenarios with large data streams. The performance of distributed DL training techniques faces the challenge of communication costs, which is vital to end devices and edge nodes. Thus, it is necessary to reduce the frequency of communication or the size of communicated data, and the corresponding methods are provided in the following section.

1) Frequency of Training Updates: In distributed DL training, a locally trained model or preprocessed data must be uploaded to a central server. One important issue is to optimize the gradient of the shared model through the gradient updates on end devices. Stochastic gradient descent (SGD) is a widely used gradient descent method that updates the minibatch gradient over the entire data set. Generally, there are two kinds of SGD: 1) synchronous and 2) asynchronous SGD [173]. In synchronous SGD, each device updates its parameters in lockstep, if all the devices finish the computation tasks, with the gradients on their local training data. Each device updates its parameters to the central server in asynchronous SGD after computing the gradients. Synchronous SGD may converge to a good solution but is slow in practice because of the requirement of waiting for the other devices. Asynchronous SGD converges faster than synchronous SGD at the cost of additional noise but updates the parameters by using stale information, thereby leading to convergence to poor solutions. Thus, more studies focus on making synchronous SGD faster or allowing asynchronous SGD to converge to better solutions. The elastic averaging SGD (EASGD) method [174] is introduced to reduce the communication costs of asynchronous SGD training methods; the basic idea is to allow each end device to carry out more local training computations and to deviate further from the central shared parameters before synchronizing its updates. This method decreases the amount of communication between local devices and central server.

FL also adopts SGD. In [169], the federated averaging (FedAvg) method for FL with a DNN established based on iterative model averaging is presented, in which end devices update the DNN model with one-step SGD and the server averages the obtained models with weights. This approach can be also applied to unbalanced and non-IID distributions. To reduce the communication costs from end devices to the central server, structured updates and sketched updates [168] are proposed and shown to enhance the efficiency of communication. The structured update method obtains the model update by restricting the parameters by using a smaller number of

TABLE IV
SUMMARY OF AIOT MODEL DISTRIBUTED TRAINING TECHNOLOGIES

Architecture	Technology	Highlights	Ref.
Distributed training	Federated Learning	• The data remains in the original position, and the optimized model is trained on user side by parameter exchange under an encryption mechanism, thus solving the problem of data islands, ensuring privacy and reducing the communication costs.	[168]–[170]
	DNN Splitting	• The deep network is split into a series of subnetworks, which are trained using partially processed data.	[171]
	Transfer Learning	• Enables a system to recognize and apply the knowledge and skills learned in previous domains/tasks to novel domains/tasks by transferring labeled data or knowledge structures from related fields	[172]

variables, as in the low-rank and random mask methods. In the sketched update process, a full model update is obtained and compressed by combined operations of subsampling, probabilistic quantization and structured random rotation before being sent to the server. To overcome the conventional shortcomings of synchronous and asynchronous SGD, straggler effects are mitigated in synchronous stochastic optimization with backup workers [173]. The core idea is to compute the minibatch gradient updates of the decentralized devices with only a subset of the worker machines. The training process updates to the full parameters and goes on to the next iteration if the server receives gradient updates from enough devices. This approach not only reduces the long latencies involved in waiting for straggler devices in synchronous SGD but also avoids asynchronous noise when adjusting for the worst stragglers.

2) Size of Training Updates: In the scenario of AIoT applications, communication bandwidth is required for gradient exchange, and the practical bandwidth limits the scalability of multinode training. In addition to the factor of the frequency of training updates, the size of training updates significantly influences on the transmission bandwidth. Gradient compression is usually adopted to reduce the size of model updates communicated to the central server, which aims to compress the gradient information of the updated model. Generally, gradient quantization and gradient sparsification are recommended to perform gradient compression [175].

Gradient quantization carries out lossy compression of the gradient vectors by using a finite-bit low-width number instead of the original floating-point gradients, which is similar to parameter quantization of inference. The difference between these methods lies in whether the quantization technique is applied to the model gradients or the model parameters. Most of the gradient exchange in distributed SGD has been proven to be redundant; thus, deep gradient compression (DGC) is proposed to decrease the communication bandwidth by compressing the updated gradients for a wide range of CNNs and RNNs [175]. To preserve the training accuracy after compression, DGC uses four techniques: 1) momentum correction; 2) local gradient clipping; 3) momentum factor masking; and 4) warm-up training.

Gradient sparsification reduces the communication costs by dropping important gradient updates and transmitting only updates that exceed a certain threshold. A convex optimization formulation [176] is employed to reduce the coding length of stochastic gradients, where the coordinates of the stochastic gradient vectors are randomly reduced and the remaining coordinates are appropriately amplified to keep the results unbiased.

Furthermore, more research has explored combinations of gradient quantization and sparsification. The convergence rate of distributed SGD for nonconvex stochastic optimization is considered using sparse parameter averaging and gradient quantization [177]. Adaptive residual gradient compression (AdaComp) is proposed to compress the updates communicated to the server by applying the gradient sparsification method in combination with efficient gradient selection and learning rate modulation [178].

C. Security Enhancement

Distributed learning has shown a strong trend toward large-scale model training in AIoT, where a server coordinates the computational power of end devices by sharing the trained data or aggregating local models trained on individual devices. This kind of method can stop privacy leaks from directly sharing the raw data collected from end devices; however, the gradient information shared by end devices still inevitably divulges private information. Thus, research and development of privacy preservation for AI model training is necessary. Generally, there are two obstacles: the first is that attackers may infer sensitive information from aggregated data or gradients; the other obstacle is that third parties are not trusted, thus causing data or model leakage. Next, this article introduces privacy-enhancing techniques from two perspectives: 1) data privacy and 2) security as well as system security.

1) Data Privacy and Security: The distributed collaborative methods used to train AI models are usually vulnerable to model poisoning attacks [179], and the client's contribution during training and sensitive information provided are susceptible to leakage by analyzing the locally trained model. To protect the privacy of the updated local models, LDP is incorporated into the local gradient descent training scheme of FL [180]. A randomly distributed update scheme is also employed to remove the security threats posed by a centralized curator. Furthermore, differential privacy is used in training an FL model by injecting Laplacian noise into the features extracted by the designed CNN model on the mobile device so that an adversary could not infer sensitive information from the learned model [181].

Cryptographic technology is also suitable for distributed model training. A multikey-based distributed DL framework is introduced to enhance data security of clients with the aid of homomorphic reencryption in asynchronous SGD [182].

Although the proposed framework adds additional communication costs, it still achieves better security properties.

Adding too much noise to the model parameter data will lead to poor performance in model training, while the sole usage of secure multiparty computation in FL will result in vulnerability to inference attacks. A hybrid approach using differential privacy and secure multiparty computation for FL systems is proposed to balance these tradeoffs, which reduces the growth of noise injection as the number of parties increases [183]. This system is suitable for various AI models (e.g., CNNs, DTs, SVMs, etc.) and provides privacy guarantees with high accuracy.

2) System Security: In the distributed training mode, a third party may use the clients' data illegally or be vulnerable to hacking, thus causing unpredictable security issues [181]. Blockchain is a decentralized and distributed shared ledger and database that stores data and features traceability, consensus, transparency and fast settlement [184]. These characteristics lay a solid foundation for creating trust in the blockchain. Many works have explored the use of blockchain to improve data security in distributed model training. To guarantee the credibility of the third party, the blockchain instead of the centralized aggregator is employed in FL, where customers sign the hashes of encrypted models and send the locally trained models to the blockchain. The miners check the identities of the senders and create a federated model by using the downloaded locally trained models [181]. One miner is chosen as the temporary leader and takes responsibility for encrypting and returning the final model to the blockchain. Such a blockchain system can protect the security of model updates by tracing malicious model updates. Lu *et al.* [185] presented a blockchain-powered differentially private data-sharing model for multiple distributed parties, where the mapped data models are shared between the multiple parties according to blockchain; consequently, the data are protected, and the trained model accuracy is enhanced.

VII. Open Challenges and Future Directions

The AIoT has a myriad of applications in daily life and brings considerable convenience to humans, however, it is still in its infancy and has broad prospects. The AIoT also faces inevitable challenges in practical deployment, including finding a cooperative mode among end devices, edge servers, and the cloud. In this section, some of the open challenges and potential future directions in AIoT are discussed.

A. Heterogeneity and Interoperability

End devices in the perception layer of the AIoT vary from the Raspberry Pi to FPGA-based products to smart phones. Owing to the diversity of sensors and devices as well as the complexity of the physical environment required to be sensed, varieties of devices need to be deployed on discrete servers for different applications or services to realize a comprehensive perception of the environment, which indicates the heterogeneity of the AIoT architecture. For example, sensing devices for AD are decorated on RSUs, and sensors for smart homes are deployed on smart gateways. To make smart decisions, data exchange and fusion among such interconnected devices are also required using heterogeneous networks, namely, Bluetooth, NB-IoT, ZigBee, Wi-Fi, the hypertext transfer protocol/transmission control protocol (HTTP/TCP) or the user datagram protocol (UDP). Thus, AIoT systems are expected to be extremely heterogeneous in terms of devices, platforms, and frameworks. Interoperability and coordination among heterogeneous devices and platforms are essential. Attempts to explore network softwarization paradigms, such as SDN [186] and network function virtualization (NFV) [187], may bring many improvements by promoting efficient and flexible operation. SDN technologies can simplify management systems by using their programmability and offer a unified framework to manage various end devices or sensors. SDN can virtualize physical devices or provide customized services to address the heterogeneity of the devices. NFV virtualizes network node functions into software modules through its virtualization technology. Recently, attempts to amalgamate SDN and NFV in edge-cloud computing have been made to improve the QoS for AIoT-driven applications [188], [189]. Furthermore, a standard communication protocol for such heterogeneous devices and sensors in the edge-cloud environment is required to support smooth communication in the network layer. OpenFlow is the standard communication protocol between an SDN controller and a switch and has attracted escalating attention from researchers [190].

DL models can be accelerated if they are implemented on a GPU edge server; however, it remains a challenge to make NFV compatible with a GPU. There is also some future work required to successfully deploy such paradigms, such as in security, resource allocation, runtime service deployment, and computational offloading.

B. Resource Management

Advancements in AIoT have created many applications, such as smart homes and the IoV. In AIoT systems, many sensors and devices are deployed distributively to collect data. Distributed sensors and devices are usually powered by batteries with limited computation and storage capacities, and it is difficult to execute latency-sensitive computation tasks on their own computing resources. To fully explore the potential of dispersive resources across edge nodes and devices, it is helpful to partition complex AI models into small subtasks and offload these subtasks to various edge nodes and devices for collaborative training. The service environments of many complex and diverse AIoT applications, such as the IoV, are highly dynamic, thus making it difficult to predict what will happen. Thus, the ability to perform online edge resource orchestration and provisioning is required to support substantial AIoT tasks. Schemes aimed at the real-time joint optimization of heterogeneous end devices' computing, communication, and caching resource coordination at runtime according to different task requirements should be thoroughly addressed. A joint caching and computing policy is designed to minimize the bandwidth cost of a wireless multicast channel [191]. Other researches aim to manage resource allocation and scheduling using AI techniques, such as DRL [43], [192].

C. Model Inference and Training

Section V presents AI inference compression and acceleration technologies involving many hyperparameters, which require empirical experiments and expert knowledge to adjust the networks. Moreover, the networks need to be retrained by fine-tuning based on many experiments. It is worthwhile to develop adaptive or automatic compression and acceleration techniques. Some studies are working in this direction [193], [194]. From a software perspective, acceleration technologies, such as pruning and quantization, may make AI models hardware friendly but decrease performance. It is a promising direction to support the execution of AI models by hardware acceleration. More hardware measures should be taken to address this problem.

Due to the limited computation, storage and network resources and the distributed heterogeneous data, it is difficult to train AI models in parallel. FL, a distributed computing architecture with benefits of little data communication traffic, lossless model quality and data isolation, breaks through the bottlenecks of data-driven requirements and privacy protection faced by AI models. However, the bandwidth of edge nodes is limited and heterogeneous, and their computing capabilities are different. In addition, different edge nodes have different amounts of data, which are unevenly distributed. Distributed SGD often results in communication delays because edge servers have to wait for all the model parameters to be returned from the host server in every iteration. A variety of parallel communication mechanisms need to be further explored to improve efficiency [195]. In addition, the existing quantization methods are generally applied in AI inference. Fine-grained quantization-aware training, which supports forward and backward propagation, can be applied to AIoT applications [196].

D. Security and Privacy

Although the AIoT brings great convenience, it is subject to security and privacy issues, such as malicious attacks and privacy leakage. As mentioned above, AI models are more likely to be deployed at the edge of the network or directly on an end device to deliver near-zero-delay services. Edge servers and end devices are often equipped with limited computation and storage resources and are confront with malicious attacks, namely, Distributed-Denial-of-Service (DDoS) attacks (such as Mirai). Flooding-based DDoS can still work effectively in edge computing systems. The excessively great computation and communication loads produced by the existing security methods are infeasible for end devices to use to protect their security. To address this issue, researchers can further emphasize lightweight security mechanisms for resource-constrained devices. Physical unclonable functions (PUFs), with the advantages of antiphysical intrusion capabilities, lower computational consumption, less resource usage, convenient implementation and unique physical attributes, will be promising research points for security authentication in edge computing environments. Furthermore, it is necessary to explore hardware-assisted protection mechanisms based on RISC-V. Thus far, some explorations have begun [197], [198].

Privacy issues are also vitally important. The AIoT may be vulnerable to data and firmware attacks. A mass of data is generated by end users and devices and is stored in local devices or edge servers, which may contain sensitive information (e.g., user location information and health or activity records). Exposure of such information will result in serious consequences. In addition, the AIoT faces transmission attacks because a mass of diverse data is required to design and train AI algorithms. The performance of algorithms will decrease if sufficient training data are not provided. Thus, data transmission between edge infrastructures will also cause privacy leakage. One promising direction is to adopt the FL method to perform privacy-preserving distributed data training. Other technologies, such as differential privacy, homomorphic encryption, and secure multiparty computation, are also employed to design parameter-sharing AI models with privacy assurance in the edge computing environment. Moreover, blockchain also plays an important role in security and privacy for the IoT, which can be combined with the foremention techniques, such as FL, to preserve privacy. However, blockchain technology developed for IoT networks fails in performance because of a waste of network resources (such as communication bandwidth and computational resources). Some researchers also resort to combining blockchain with the IoT in the context of the sixth-generation (6G) communication network to reduce computational cost, which shows great potentials in AI and data storage and analytics [199]. Thus, blockchain-based solutions for the AIoT should be further explored to protect users and devices from attacks.

E. Artificial Intelligence Ethics in Artificial Intelligence of Things

In AIoT systems, AI decision supports may be made in milliseconds with no time for human oversight, which requires AI algorithms to autonomously learn in real-world scenarios without harming people or undermining their rights. To address the ethical problems of AI-based technology, some design principles, e.g., justice, honesty, accountability, safety and sustainability, need to considered. In AI ethics, justice that refrains from having any prejudice or favoritism toward individuals is prioritized the highest. The realization of justice consists of the fairness, nondiscrimination and diversity of data, algorithm, implementation and outcome. Honesty, the core of fulfilling the series of AI ethical issues, underpins the idea of explainability or technical transparency regarding AI systems. Achieving AI honesty requires transparency, openness and interpretability of data and technology, and acknowledging errors. Accountability cannot be neglected in AI ethics, where AI developers, designers or institutions should be accountable for AI actions and their outcomes. Accountability should be established across the whole design and implementation workflow. Safety is the destination of AI ethics, which care more about the accuracy, reliability, security, and robustness of AI systems. To enhance security, AI designers should state that AI systems will never result in foreseeable or unintentional harm, including military war or malicious cyber hacking. Finally, sustainability in AI ethics requires protecting the environment

and improving the ecosystem when developing and deploying AI systems. To reach their goal, AI-based applications should be designed, deployed and managed considering increases in their energy efficiencies and reductions in their ecological footprints.

VIII. Conclusion

The extensive use of the IoT faces a series of challenges, including data explosion and heterogeneity. The emergence of advanced AI technologies is a potential solution to these challenges and can help the IoT discover significant information, make accurate predictions and carry out early responses. The end-edge-cloud coordination AIoT provides flexible QoS guarantees according to various requirements, such as services in different delay ranges and high-accuracy predictions, and enables multiple users to participate. However, the edge-cloud coordination AIoT may also endanger human life when encountering faults and situations that have never occurred or improper operation. Moreover, with various household appliances (smart lights, cars, etc.) accessing the Internet, on the one hand, life becomes increasingly convenient; on the other hand, it is more convenient for hackers to attack. Additionally, people may become lazy due to enjoying such conveniences. AIoT applications can replace jobs due to their low cost and high efficiency, thereby leading to fewer job opportunities. The AIoT also cannot discover the root causes of errors that are difficult to correct and need to be modified manually.

This survey explores the edge-computing-enabled integration of AI with the IoT. This article comprehensively reviews the current research efforts on AIoT. Specifically, an overview of the IoT, AI technology and edge computing is given. Then, this article discusses opportunities for the synthesis of IoT and AI, depicts a general AIoT architecture and introduces a practical AIoT example to explain how AI can be applied in real-world scenarios. Additionally, the key AIoT technologies for AI models regarding inference and training at the edge of the network are provided in detail. This article further outlines the open challenges and future directions of AIoT.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] M. Shirer and C. MacGillivray. *The Growth in Connected IoT Devices Is Expected to Generate 79.4 zb of Data in 2025, According to a New IDC Forecast*. Accessed: 2019. [Online]. Available: https://www.iotcentral.io/blog/iot-is-not-a-buzzword-but-necessity

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

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[5] B. Heintz, A. Chandra, and R. K. Sitaraman, "Optimizing grouped aggregation in geo-distributed streaming analytics," in *Proc. 24th Int. Symp. High Perform. Parallel Distrib. Comput.*, 2015, pp. 133–144.

[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[7] 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.

[8] W. Hu *et al.*, "Quantifying the impact of edge computing on mobile applications," in *Proc. 7th ACM SIGOPS Asia–Pac. Workshop Syst.*, 2016, pp. 1–8.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[10] F. Bu and X. Wang, "A smart agriculture IoT system based on deep reinforcement learning," *Future Gener. Comput. Syst.*, vol. 99, pp. 500–507, Oct. 2019.

[11] W. Zhan *et al.*, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[12] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for Internet of Vehicles," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 12–18, Jun. 2019.

[13] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[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] A. Ghosh, D. Chakraborty, and A. Law, "Artificial intelligence in Internet of Things," *CAAI Trans. Intell. Technol.*, vol. 3, no. 4, pp. 208–218, 2018.

[16] I. U. Din, M. Guizani, J. J. Rodrigues, S. Hassan, and V. V. Korotaev, "Machine learning in the Internet of Things: Designed techniques for smart cities," *Future Gener. Comput. Syst.*, vol. 100, pp. 826–843, Nov. 2019.

[17] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for Internet of Things," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 8, pp. 1399–1417, 2018.

[18] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[19] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions," *ACM Comput. Surveys*, vol. 53, no. 4, pp. 1–37, 2020.

[20] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[21] 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.

[22] A. H. Sodhro, S. Pirbhulal, and V. H. C. De Albuquerque, "Artificial intelligence-driven mechanism for edge computing-based industrial applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4235–4243, Jul. 2019.

[23] H. Ji, O. Alfarraj, and A. Tolba, "Artificial intelligence-empowered edge of vehicles: Architecture, enabling technologies, and applications," *IEEE Access*, vol. 8, pp. 61020–61034, 2020.

[24] S. Sankaranarayanan and S. Mookherji, "SVM-based traffic data classification for secured IoT-based road signaling system," in *Research Anthology on Artificial Intelligence Applications in Security*. Hershey, PA, USA: IGI Global, 2021, pp. 1003–1030.

[25] D. Valluru and I. J. S. Jeya, "IoT with cloud based lung cancer diagnosis model using optimal support vector machine," *Health Care Manag. Sci.*, vol. 23, no. 4, pp. 670–679, Dec. 2020.

[26] S. Panda and G. Panda, "Intelligent classification of IoT traffic in healthcare using machine learning techniques," in *Proc. IEEE 6th Int. Conf. Control Autom. Robot. (ICCAR)*, 2020, pp. 581–585.

[27] A. Alabdulkarim, M. Al-Rodhaan, T. Ma, and Y. Tian, "Ppsdt: A novel privacy-preserving single decision tree algorithm for clinical decision-support systems using IoT devices," *Sensors*, vol. 19, no. 1, p. 142, 2019.

[28] X. Wang, C. Shao, S. Xu, S. Zhang, W. Xu, and Y. Guan, "Study on the location of private clinics based on k-means clustering method and an integrated evaluation model," *IEEE Access*, vol. 8, pp. 23069–23081, 2020.

[29] X. Liu, G. Chen, X. Sun, and A. Knoll, "Ground moving vehicle detection and movement tracking based on the neuromorphic vision sensor," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9026–9039, Sep. 2020.

[30] K. Costa *et al.*, "Comparison of the techniques decision tree and MLP for data mining in spams detection to computer networks," in *Proc. IEEE 3rd Int. Conf. Innov. Comput. Technol. (INTECH)*, 2013, pp. 344–348.

[31] A. Mukherjee, S. Misra, N. S. Raghuwanshi, and S. Mitra, "Blind entity identification for agricultural IoT deployments," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3156–3163, Apr. 2019.

[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[33] S. Khan, K. Muhammad, S. Mumtaz, S. W. Baik, and V. H. C. de Albuquerque, "Energy-efficient deep CNN for smoke detection in foggy IoT environment," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9237–9245, Dec. 2019.

[34] V. Bianchi, M. Bassoli, G. Lombardo, P. Fornacciari, M. Mordonini, and I. De Munari, "IoT wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8553–8562, Oct. 2019.

[35] G. Bedi, G. K. Venayagamoorthy, and R. Singh, "Development of an IoT-driven building environment for prediction of electric energy consumption," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4912–4921, Jun. 2020.

[36] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Sci. Rep.*, vol. 8, no. 1, pp. 1–12, 2018.

[37] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "SR-LSTM: State refinement for LSTM towards pedestrian trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12085–12094.

[38] H. Zheng, F. Lin, X. Feng, and Y. Chen, "A hybrid deep learning model with attention-based Conv-LSTM networks for short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 9, 2020, doi: 10.1109/TITS.2020.2997352.

[39] I. J. Goodfellow *et al.*, "Generative adversarial networks," 2014. [Online]. Available: arXiv:1406.2661.

[40] A. Natani, A. Sharma, T. Peruma, and S. Sukhavasi, "Deep learning for multi-resident activity recognition in ambient sensing smart homes," in *Proc. IEEE 8th Global Conf. Consum. Electron. (GCCE)*, 2019, pp. 340–341.

[41] P. Wei, S. Xia, R. Chen, J. Qian, C. Li, and X. Jiang, "A deep-reinforcement-learning-based recommender system for occupant-driven energy optimization in commercial buildings," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6402–6413, Jul. 2020.

[42] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[43] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[44] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double *Q*-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016, pp. 2094–2100.

[45] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks," *IEEE Trans. Mobile Comput.*, early access, Sep. 1, 2020, doi: 10.1109/TMC.2020.3017079.

[46] Y. Liang, C. Guo, Z. Ding, and H. Hua, "Agent-based modeling in electricity market using deep deterministic policy gradient algorithm," *IEEE Trans. Power Syst.*, vol. 35, no. 6, pp. 4180–4192, Nov. 2020.

[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: arXiv:1707.06347.

[48] M. A Vouk, "Cloud computing—Issues, research and implementations," *J. Comput. Inf. Technol.*, vol. 16, no. 4, pp. 235–246, 2008.

[49] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 133–143, 2014.

[50] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[51] F. Bonomi, R. A. 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.

[52] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Services*, 2012, pp. 29–36.

[53] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[54] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Cham, Swizerland: Springer, 2014, pp. 169–186.

[55] G. Mei, N. Xu, J. Qin, B. Wang, and P. Qi, "A survey of Internet of Things (IoT) for geohazard prevention: Applications, technologies, and challenges," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4371–4386, May 2020.

[56] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[57] E. TPU. *Google's Purpose-Built ASIC Designed to Run Inference at the Edge*. Accessed: 2020. [Online]. Available: https://cloud.google.com/edge-tpu/

[58] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam, "DianNao family: Energy-efficient hardware accelerators for machine learning," *Commun. ACM*, vol. 59, no. 11, pp. 105–112, 2016.

[59] NVIDIA. *Turing GPU Architecture*. Accsesed: 2018. [Online]. Available: https://www.nvidia.com/en-us/geforce/turing/

[60] NVIDIA. *Adaptable Intelligent*. Accessd: 2020. [Online]. Available: https://www.xilinx.com/products/silicon-devices/cost-optimized-portfolio.html

[61] HiSilicon. *The World's First Full-Stack All-Scenario Ai Chip*. Accessed: 2020. [Online]. Available: http://www.hisilicon.com/en/Products/ProductList/Ascend

[62] Samsung. *Mobile Processor Exynos 9820*. Accessed: 2020. [Online]. Available: https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-9-series-9820/

[63] Intel. *Advanced Intelligence for High-Density Edge Solution*. Accessed: 2020. [Online]. Avilable: https://www.intel.com/content/www/us/en/products/docs/processors/xeon/d-2100-brief.html

[64] J. Hsu, "IBM's new brain [news]," *IEEE Spectr.*, vol. 51, no. 10, pp. 17–19, Sep. 2014.

[65] Open Networking Foundation. *Cord*. Accessed: 2018. [Online]. Available: https://www.opennetworking.org/cord

[66] T. L. Foundation. *Edgex Foundry*. Accessed: 2018. [Online]. Available: https://www.edgexfoundry.org

[67] T. L. Foundation. *Akraino Edge Stack*. Accessed: 2018. [Online]. Available: https://www.lfedge.org/projects/akraino/

[68] M. Azure. *Azure IoT Edge, Extend Cloud Intelligence and Analytics to Edge Devices*. Accessed: 2018. [Online]. Available: https://github.com/Azure/iotedge

[69] A. W. Services. *AWS IoT Greengrass*. Accessed: 2019. [Online]. Available: https://aws.amazon.com/cn/greengrass/

[70] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, 2018, pp. 373–377.

[71] OpenEdge. *Extend Cloud Computing, Data and Service Seamlessly to Edge Devices*. Accessed: 2020. [Online]. Available: https://github.com/baetyl/baetyl

[72] Q. Zhang *et al.*, "OpenVDAP: An open vehicular data analytics platform for CAVs," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2018, pp. 1310–1320.

[73] C.-C. Hung *et al.*, "VideoEdge: Processing camera streams using hierarchical clusters," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, 2018, pp. 115–131.

[74] Z. Du *et al.*, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, 2015, pp. 92–104.

[75] Nvidia Corporation. *Jetson TX2 Module*. Accessed: 2019. [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx2

[76] S. Kato *et al.*, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proc. ACM/IEEE 9th Int. Conf. Cyber Phys. Syst. (ICCPS)*, 2018, pp. 287–296.

[77] Y. Wang *et al.*, "Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 623–633, May 2020.

[78] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[79] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[80] TAS Foundation. *Apache Edgent*. Accessed: 2018. [Online]. Available: https://github.com/apache/incubator-retired-edgent

[81] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.

[82] T. Wu, Y. Wang, W. Shi, and J. Lu, "HydraMINI: An FPGA-based affordable research and education platform for autonomous driving," in *Proc. IEEE Int. Conf. Connected Auton. Driving (MetroCAD)*, 2020, pp. 45–52.

[83] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*. Madison, WI, USA: Arpaci-Dusseau, 2018.

[84] Xilinx Inc. *DNNDK: Deep Neural Network Development Kit*. Accessed: 2019. [Online]. Available: https://www.xilinx.com/products/design-tools/ai-inference/edgeai-platform.htmldnndk

[85] Y. Wang, L. Liu, X. Zhang, and W. Shi, "HydraOne: An indoor experimental research and education platform for CAVs," in *Proc. 2nd {USENIX} Workshop Hot Topics Edge Comput. (HotEdge)*, 2019, pp. 1–7.

[86] 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 INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2019, pp. 1–6.

[87] A. Villanueva, R. L. L. Benemerito, M. J. M. Cabug-Os, R. B. Chua, C. K. D. Rebeca, and M. Miranda, "Somnolence detection system utilizing deep neural network," in *Proc. IEEE Int. Conf. Inf. Commun. Technol. (ICOIACT)*, 2019, pp. 602–607.

[88] W.-J. Tsaur and L.-Y. Yeh, "DANS: A secure and efficient driver-abnormal notification scheme with IoT devices over IoV," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1628–1639, Jun. 2019.

[89] D. Tian, C. Zhang, X. Duan, and X. Wang, "An automatic car accident detection method based on cooperative vehicle infrastructure systems," *IEEE Access*, vol. 7, pp. 127453–127463, 2019.

[90] J. Berrezueta-Guzman, I. Pau, M.-L. Martín-Ruiz, and N. Máximo-Bocanegra, "Smart-home environment to support homework activities for children," *IEEE Access*, vol. 8, pp. 160251–160267, 2020.

[91] F. Wang, W. Gong, and J. Liu, "On spatial diversity in WiFi-based human activity recognition: A deep learning-based approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2035–2047, Apr. 2019.

[92] H. Zou, Y. Zhou, H. Jiang, S.-C. Chien, L. Xie, and C. J. Spanos, "WinLight: A WiFi-based occupancy-driven lighting control system for smart building," *Energy Build.*, vol. 158, pp. 924–938, Jan. 2018.

[93] Q. Shi *et al.*, "Deep learning enabled smart mats as a scalable floor monitoring system," *Nat. Commun.*, vol. 11, no. 1, pp. 1–11, 2020.

[94] F. De Vita, G. Nocera, D. Bruneo, V. Tomaselli, D. Giacalone, and S. K. Das, "Quantitative analysis of deep leaf: A plant disease detector on the smart edge," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, 2020, pp. 49–56.

[95] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-time apple detection system using embedded systems with hardware accelerators: An edge Ai application," *IEEE Access*, vol. 8, pp. 9102–9114, 2020.

[96] C. Krintz, R. Wolski, N. Golubovic, and F. Bakir, "Estimating outdoor temperature from CPU temperature for IoT applications in agriculture," in *Proc. 8th Int. Conf. Internet Things*, 2018, pp. 1–8.

[97] G. Lavanya, C. Rani, and P. Ganeshkumar, "An automated low cost IoT based fertilizer intimation system for smart agriculture," *Sustain. Comput. Informat. Syst.*, vol. 28, Dec. 2020, Art. no. 100300.

[98] S. K. Singh, F. Carpio, and A. Jukan, "Improving animal-human cohabitation with machine learning in fiber-wireless networks," *J. Sensor Actuator Netw.*, vol. 7, no. 3, p. 35, 2018.

[99] R. Nikhil, B. Anisha, and R. Kumar, "Real-time monitoring of agricultural land with crop prediction and animal intrusion prevention using Internet of Things and machine learning at edge," in *Proc. IEEE Int. Conf. Electron. Comput. Commun. Technol. (CONECCT)*, 2020, pp. 1–6.

[100] A. Ahad, M. Tahir, and K.-L. A. Yau, "5G-based smart healthcare network: Architecture, taxonomy, challenges and future research directions," *IEEE Access*, vol. 7, pp. 100747–100762, 2019.

[101] Z. Zhang, T. He, M. Zhu, Q. Shi, and C. Lee, "Smart triboelectric socks for enabling Artificial Intelligence of Things (AIoT) based smart home and healthcare," in *Proc. IEEE 33rd Int. Conf. Micro Electro Mech. Syst. (MEMS)*, 2020, pp. 80–83.

[102] S. Tuli *et al.*, "HealthFog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments," *Future Gener. Comput. Syst.*, vol. 104, pp. 187–200, Mar. 2020.

[103] S. K. Sood and I. Mahajan, "A fog-based healthcare framework for Chikungunya," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 794–801, Apr. 2018.

[104] M. S. Hossain, G. Muhammad, and N. Guizani, "Explainable AI and mass surveillance system-based healthcare framework to COMBAT COVID-I9 like pandemics," *IEEE Netw.*, vol. 34, no. 4, pp. 126–132, Jul./Aug. 2020.

[105] Y. Liu, Z. Zhao, F. Chang, and S. Hu, "An anchor-free convolutional neural network for real-time surgical tool detection in robot-assisted surgery," *IEEE Access*, vol. 8, pp. 78193–78201, 2020.

[106] H. Qiu, Z. Li, Y. Yang, C. Xin, and G.-B. Bian, "Real-time iris tracking using deep regression networks for robotic ophthalmic surgery," *IEEE Access*, vol. 8, pp. 50648–50658, 2020.

[107] M. M. Hosseini, A. Umunnakwe, M. Parvania, and T. Tasdizen, "Intelligent damage classification and estimation in power distribution poles using unmanned aerial vehicles and convolutional neural networks," *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 3325–3333, Jul. 2020.

[108] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2505–2516, Sep. 2017.

[109] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, "Online cyber-attack detection in smart grid: A reinforcement learning approach," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5174–5185, Sep. 2019.

[110] Y. Wang, Y. Shen, S. Mao, X. Chen, and H. Zou, "LASSO and LSTM integrated temporal model for short-term solar intensity forecasting," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2933–2944, Apr. 2019.

[111] M. D'Incecco, S. Squartini, and M. Zhong, "Transfer learning for non-intrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1419–1429, Mar. 2020.

[112] Z. Wan, H. Li, H. He, and D. Prokhorov, "Model-free real-time EV charging scheduling based on deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5246–5257, Sep. 2019.

[113] Y.-W. Lee, "A stochastic model of particulate matters with AI-enabled technique-based IoT gas detectors for air quality assessment," *Microelectron. Eng.*, vol. 229, May 2020, Art. no. 111346.

[114] X. Sun, X. Wang, D. Cai, Z. Li, Y. Gao, and X. Wang, "Multivariate seawater quality prediction based on PCA-RVM supported by edge computing towards smart ocean," *IEEE Access*, vol. 8, pp. 54506–54513, 2020.

[115] J. Gotthans, T. Gotthans, and R. Marsalek, "Deep convolutional neural network for fire detection," in *Proc. IEEE 30th Int. Conf. Radioelektronika (RADIOELEKTRONIKA)*, 2020, pp. 1–6.

[116] D. Kinaneva, G. Hristov, J. Raychev, and P. Zahariev, "Early forest fire detection using drones and artificial intelligence," in *Proc. IEEE 42nd Int. Convent. Inf. Commun. Technol. Electron. Microelectron. (MIPRO)*, 2019, pp. 1060–1065.

[117] K. Ahmad, K. Khan, and A. Al-Fuqaha, "Intelligent fusion of deep features for improved waste classification," *IEEE Access*, vol. 8, pp. 96495–96504, 2020.

[118] T. J. Sheng *et al.*, "An Internet of Things based smart waste management system using LoRa and TensorFlow deep learning model," *IEEE Access*, vol. 8, pp. 148793–148811, 2020.

[119] L. Hu, Y. Miao, G. Wu, M. M. Hassan, and I. Humar, "iRobot-factory: An intelligent robot factory based on cognitive manufacturing and edge computing," *Future Gener. Comput. Syst.*, vol. 90, pp. 569–577, Jan. 2019.

[120] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4665–4673, Oct. 2018.

[121] Y. Feng, T. Wang, B. Hu, C. Yang, and J. Tan, "An integrated method for high-dimensional imbalanced assembly quality prediction supported by edge computing," *IEEE Access*, vol. 8, pp. 71279–71290, 2020.

[122] K. S. Kiangala and Z. Wang, "An effective predictive maintenance framework for conveyor motors using dual time-series imaging and convolutional neural network in an industry 4.0 environment," *IEEE Access*, vol. 8, pp. 121033–121049, 2020.

[123] 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.

[124] D. Yao, M. Wen, X. Liang, Z. Fu, K. Zhang, and B. Yang, "Energy theft detection with energy privacy preservation in the smart grid," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7659–7669, Oct. 2019.

[125] TL Foundation. *LF Energy GXF*. Accessed: 2020. [Online]. Available: https://www.lfenergy.org/projects/gxf/

[126] F. Ma, X. Luo, and E. Litvinov, "Cloud computing for power system simulations at ISO New England—Experiences and challenges," *IEEE Trans. Smart Grid*, vol. 7, no. 6, pp. 2596–2603, Nov. 2016.

[127] Ž. N. Popović, B. B. Radmilović, and V. M. Gačić, "Smart grids concept in electrical distribution system," *Ther. Sci.*, vol. 16, no. s1, pp. 205–213, 2012.

[128] S. Chen *et al.*, "Internet of Things based smart grids supported by intelligent edge computing," *IEEE Access*, vol. 7, pp. 74089–74102, 2019.

[129] PNN Laboratory. *Transactive Energy: Negotiating New Terrain*. Accessed: 2020. [Online]. Available: https://www.pnnl.gov/news-media/transactive-energy-negotiating-new-terrain

[130] Z. Yang and D. Li, "WasNet: A neural network-based garbage collection management system," *IEEE Access*, vol. 8, pp. 103984–103993, 2020.

[131] 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. [Online]. Available: arXiv:1602.07360.

[132] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[133] 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.*, 2018, pp. 6848–6856.

[134] W. Ding, Z. Huang, Z. Huang, L. Tian, H. Wang, and S. Feng, "Designing efficient accelerator of depthwise separable convolutional neural network on FPGA," *J. Syst. Architect.*, vol. 97, pp. 278–286, Aug. 2019.

[135] J. Redmon, S. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[136] S. Han, H. Mao, and W. Dally, "Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015. [Online]. Available: arxiv.abs/1510.00149

[137] X. Liu, J. Pool, S. Han, and W. J. Dally, "Efficient sparse-winograd convolutional neural networks," 2018. [Online]. Available: arXiv:1802.06367.

[138] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5058–5066.

[139] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2736–2744.

[140] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1389–1397.

[141] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, 2014, pp. 1–6.

[142] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2017, pp. 15–24.

[143] J. Qiu *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2016, pp. 26–35.

[144] Y. H. Oh *et al.*, "A portable, automatic data qantizer for deep neural networks," in *Proc. 27th Int. Conf. Parallel Architect. Compilation Techn.*, 2018, pp. 1–14.

[145] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: arXiv:1503.02531.

[146] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014. [Online]. Available: arXiv:1412.6550.

[147] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015. [Online]. Available: arXiv:1510.00149.

[148] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du, "On-demand deep model compression for mobile devices: A usage-driven model selection framework," in *Proc. 16th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2018, pp. 389–400.

[149] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 1421–1429.

[150] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "MCDNN: An approximation-based execution framework for deep stream processing under resource constraints," in *Proc. 14th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2016, pp. 123–136.

[151] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Services*, 2010, pp. 49–62.

[152] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[153] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proc. 13th ACM Conf. Embedded Netw. Sensor Syst.*, 2015, pp. 155–168.

[154] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, 2015, pp. 9–16.

[155] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proc. 11th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2013, pp. 139–152.

[156] 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.

[157] 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.

[158] Z. Zhao, Z. Jiang, N. Ling, X. Shuai, and G. Xing, "ECRT: An edge computing system for real-time image-based object tracking," in *Proc. 16th ACM Conf. Embedded Netw. Sensor Syst.*, 2018, pp. 394–395.

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

[160] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 328–339.

[161] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, 2018, pp. 31–36.

[162] M. AboulAtta, M. Ossadnik, and S.-A. Ahmadi, "Stabilizing inputs to approximated nonlinear functions for inference with homomorphic encryption in deep neural networks," 2019. [Online]. Available: arXiv:1902.01870.

[163] Y. Li, H. Li, G. Xu, S. Liu, and R. Lu, "EPPS: Efficient privacy-preserving scheme in distributed deep learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[164] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *Proc. 27th {USENIX} Security Symp. ({USENIX} Security)*, 2018, pp. 1651–1669.

[165] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *Proc. 25th Int. Conf. Architect. Support Program. Lang. Oper. Syst.*, 2020, pp. 3–18.

[166] C. Xu, J. Ren, L. She, Y. Zhang, Z. Qin, and K. Ren, "EdgeSanitizer: Locally differentially private deep inference at the edge for mobile data analytics," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5140–5151, Jun. 2019.

[167] Z. He, T. Zhang, and R. B. Lee, "Attacking and protecting data privacy in edge-cloud collaborative inference systems," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9706–9716, Jun. 2021.

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

[169] 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.

[170] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1322–1333.

[171] 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.

[172] R. Sharma, S. Biookaghazadeh, B. Li, and M. Zhao, "Are existing knowledge transfer techniques effective for deep learning with edge devices?" in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, 2018, pp. 42–49.

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

[174] S. Zhang, A. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," 2014. [Online]. Available: arXiv:1412.6651.

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

[176] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," 2017. [Online]. Available: arXiv:1710.09854.

[177] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2530–2541.

[178] C. Hardy, E. Le Merrer, and B. Sericola, "Distributed deep learning on edge-devices: Feasibility via adaptive compression," in *Proc. IEEE 16th Int. Symp. Netw. Comput. Appl. (NCA)*, 2017, pp. 1–8.

[179] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018. [Online]. Available: arXiv:1808.04866.

[180] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2134–2143, Mar. 2020.

[181] Y. Zhao, J. Zhao, L. Jiang, R. Tan, and D. Niyato, "Mobile edge computing, blockchain and reputation-based crowdsourcing IoT federated learning: A secure, decentralized and privacy-preserving system," 2019. [Online]. Available: arXiv:1906.10893.

[182] F. Tang, W. Wu, J. Liu, H. Wang, and M. Xian, "Privacy-preserving distributed deep learning via homomorphic re-encryption," *Electronics*, vol. 8, no. 4, p. 411, 2019.

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

[184] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[185] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[186] C. Wang, Y. Zhang, X. Chen, K. Liang, and Z. Wang, "SDN-based handover authentication scheme for mobile edge computing in cyber-physical systems," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8692–8701, Oct. 2019.

[187] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, "Reliability-aware virtualized network function services provisioning in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2699–2713, Nov. 2020.

[188] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5706–5712, Jul. 2020.

[189] M. Wang, B. Cheng, W. Feng, and J. Chen, "An efficient service function chain placement algorithm in a MEC-NFV environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[190] A. Mondal, S. Misra, and I. Maity, "AMOPE: Performance analysis of OpenFlow systems in software-defined networks," *IEEE Syst. J.*, vol. 14, no. 1, pp. 124–131, Mar. 2020.

[191] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Bandwidth gain from mobile edge computing and caching in wireless multicast systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 3992–4007, Jun. 2020.

[192] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2773–2782, Dec. 2020.

[193] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "AutoCompress: An automatic DNN structured pruning framework for ultra-high compression rates," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 4876–4883.

[194] 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.

[195] D. Rothchild *et al.*, "FetchSGD: Communication-efficient federated learning with sketching," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8253–8265.

[196] C.-C. Chung, W.-T. Chen, and Y.-C. Chang, "Using quantization-aware training technique with post-training fine-tuning quantization to implement a mobilenet hardware accelerator," in *Proc. Indo–Taiwan 2nd Int. Conf. Comput. Anal. Netw. (Indo-Taiwan ICAN)*, 2020, pp. 28–32.

[197] J. Long, W. Liang, K.-C. Li, D. Zhang, M. Tang, and H. Luo, "PUF-based anonymous authentication scheme for hardware devices and IPs in edge computing environment," *IEEE Access*, vol. 7, pp. 124785–124796, 2019.

[198] A. De, A. Basu, S. Ghosh, and T. Jaeger, "Hardware assisted buffer protection mechanisms for embedded RISC-V," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4453–4465, Dec. 2020.

[199] R. Sekaran, R. Patan, A. Raveendran, F. Al-Turjman, M. Ramachandran, and L. Mostarda, "Survival study on blockchain based 6G-enabled mobile edge computation for IoT automation," *IEEE Access*, vol. 8, pp. 143453–143463, 2020.

Zhuoqing Chang (Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science, Wuhan University, Wuhan, China.

His current research interests include the Internet of Things, deep learning, and missing data.

Shubo Liu received the Ph.D. degree in communication and information system from Wuhan University, Wuhan, China, in 2009.

He is currently a Full Professor with the School of Computer Science, Wuhan University. His research interests are on multimedia information processing and security, Internet of Things, and edge computing.

Xingxing Xiong is currently pursuing the Ph.D. degree with the Department of Cyber Space Security, Wuhan University, Wuhan, China.

His main research interests include privacy protection and IoT security.

Zhaohui Cai received the Ph.D. degree from Wuhan University, Wuhan, China, in 2005.

She is currently an Associate Professor with the School of Computer Science, Wuhan University. She is mainly engaged in machine learning, Internet-of-Things technology, and big data analysis and has published many articles and participated in the National 863 program, the National Natural Science Foundation, and the provincial and ministerial scientific and technological research projects.

Guoqing Tu received the B.E. degree from the Department of Computer and Application, Hefei University of Technology, Hefei, China, in 1996, and the M.S. and Ph.D. degrees from the School of Computer Science, Wuhan University, Wuhan, China, in 2002 and 2008, respectively.

He is currently an Associated Professor with the School of Cyber Science and Engineering, Wuhan University. His current research interests include IoT and its security, embedded system, and hydro-logical monitoring system.