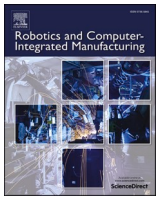
Robotics and Computer–Integrated Manufacturing 77 (2022) 102351

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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



Cloud-edge-device collaboration mechanisms of deep learning models for smart robots in mass personalization

Chen Yang [a], Yingchao Wang [a], Shulin Lan [b,*], Lihui Wang [c], Weiming Shen [d], George Q. Huang [e]

[a] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, PR China
[b] School of Economics and Management, University of Chinese Academy of Sciences, Beijing, PR China
[c] Department of Production Engineering, KTH Royal Institute of Technology, Sweden
[d] State Key Lab of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, PR China
[e] Department of Industrial and Manufacturing Systems Engineering, University of Hong Kong, Hong Kong, PR China

ARTICLE INFO

Keywords:
Cloud-edge-device collaboration
Cloud manufacturing
Smart robots
Deep learning
Mass personalization
Distributed deep learning
Collaborative learning

ABSTRACT

Personalized products have gradually become the main business model and core competencies of many enterprises. Large differences in components and short delivery cycles of such products, however, require industrial robots in cloud manufacturing (CMfg) to be smarter, more responsive and more flexible. This means that the deep learning models (DLMs) for smart robots should have the performance of real-time response, optimization, adaptability, dynamism, and multimodal data fusion. To satisfy these typical demands, a cloud-edge-device collaboration framework of CMfg is first proposed to support smart collaborative decision-making for smart robots. Meanwhile, in this context, different deployment and update mechanisms of DLMs for smart robots are analyzed in detail, aiming to support rapid response and high-performance decision-making by considering the factors of data sources, data processing location, offline/online learning, data sharing and the life cycle of DLMs. In addition, related key technologies are presented to provide references for technical research directions in this field.

1. Introduction

Mass Personalized Products (MPPs), adapted to meet individual customers' requirements and needs, have gradually become one main business model, as well as the core competitiveness of enterprises to stabilize and expand the market. It requires product fulfillment (where industrial robots play a key role) to be changeable, adaptable, and configurable, because not only the final product but also the basic design and product structure must be able to differentiate at the module and parameter level to meet individual's unique needs. As shown in Fig. 1, Mass Products (MPs) focus on the large-scale production of common and standardized products for the vast majority, while Mass Customization Products (MCPs) and MPPs form the so-called "long-tail", because they deal with the small batch production or the production of a personalized product. Driven by technology development and market requirements, MPPs can also produce substantial economic benefits when the technology is mature and the operation cost is low enough. Therefore, personalized products can fulfill customer needs for individual preferences and uniqueness, resulting in increased revenue from boosting overall sales to enhancing the average order value.

Cloud manufacturing (CMfg), which virtualizes and manages mass manufacturing resources and capabilities and provides them as manufacturing services, is a promising manufacturing mode [1–3]. However, personalized products have the characteristics of order-driven production, strict processing times, high dynamic external conditions, and considerable flexibility in the production process. These bring more uncertainties to the production system (composed of industrial robots) and significant challenges in the adaptive processing of related tasks for robots in mass personalized production. Large differences in components and processes of MPPs require that robots in CMfg can intelligently deal with individual MPPs parts, make timely adaptive decisions, and perform dynamic reconfiguration [4] to manufacture efficiently mass personalized products. For this reason, smart robots in CMfg need to have the self-X (aware, optimizing, and learning) ability to learn new methods, knowledge, and skills from their experience, collaborate with human workers, make smart decisions [5], and face complex and dynamic situations of MPPs.

By adopting a large number of hidden layers (mostly non-linear) and

* Corresponding author.
E-mail address: lanshulin@ucas.ac.cn (S. Lan).

https://doi.org/10.1016/j.rcim.2022.102351
Received 6 February 2022; Received in revised form 9 March 2022; Accepted 22 March 2022
Available online 28 March 2022

combining automatic feature engineering with the learning process, deep learning (DL) has a strong ability to learn the essential characteristics of the data from samples, thereby achieving outstanding performance. With the help of DL, robots endowed with self-X (aware, optimizing, and learning) ability become smarter. However, since DL often requires high-performance computing resources (GPUs, CPUs and storage devices) for model training and execution on massive data, exiting robots for manufacturing may not fulfill this stringent requirement on computing capability. Meanwhile, there is an imprecise trend: the more layers and parameters of a deep neural network, the more accurate the decision-making, which would undoubtedly increase the training and running cost of deep learning models (DLMs). A common practice is to outsource (or upload) the shopfloor manufacturing data to the cloud computing center, that can afford computing-intensive tasks, and send the decision result to the smart robots. As a result, with the deployment of a large number of robots and other sensor devices in smart factories, the volume of real-time data generated can reach PB or even ZB level [6] while transferring raw manufacturing data to remote clouds, which inevitably leads to high latency, data loss, and network congestion. Hence, this common cloud computing practice may fail to fulfill the real-time requirement of time-sensitive tasks for smart robots.

As a supplement to cloud computing, edge computing can provide timely computing services because of its proximity to manufacturing resources (data sources) [7]. However, its limited computing resources make it difficult to support large DLMs that have complex structures and a large number of parameters. This means that learning from a large amount of historical data still needs to be completed in the cloud center. Moreover, MPPs require precise and timely control of the movements of the end equipment. Therefore, it is necessary to migrate (part of) intelligence from the cloud center to edge and end devices (mainly industrial robots) to meet the requirements of MPPs scenarios for smart real-time response through cloud-edge-device collaboration. To make CMfg better cope with the impact of mass personalization, internal and external uncertainties, and high dynamic factors, it is essential to conduct research on the deep integration of cloud-edge-device collaboration with CMfg for smart robots.

2. Mass personalized production and deep learning

2.1. Characteristics of mass personalized production

Personalized products that can be manufactured by additive manufacturing are not the focus of this article. Instead, this paper discusses a new discrete manufacturing paradigm: personalized product realization tailored to the individual needs and preferences of consumers. A typical example is a personalized vehicle interior to match the specific needs of an individual.

With the help of pervasive connections, MPPs require customer engagement in product design and manufacturing processes, or even in the entire life cycle of the product. To produce MPPs with a wide variety and small batches in a short cycle, an open product platform is utilized to allow various modules, including user-designed modules, to be integrated. Usually, an MPP consists of three types of modules [8]: common modules that are shared across the product platform, customized modules that allow customers to choose, mix and match, and personalized modules that allow customers to create and design. The standard mechanical, electrical and informational interfaces of these modules allow easy assembly and disassembly.

To manufacture MPPs, production resources, cloud platforms, edge servers, and upper monitoring terminals should be closely connected. To support efficient manufacturing of MPPs, the CMfg system should have the ability of dynamic and rapid reconfiguration for different product variants. To make smart decisions and control the system (including system reconfiguration) for MPPs, it is necessary to have the real-time pervasive monitoring of manufacturing resources and smart data processing ability.

2.2. DL-enabled smart robots for mass personalized production

(1) Real-time smart decision-making for machine-to-machine collaboration

As shown in Fig. 2, the MPPs require the CMfg system to (1) intelligently identify different personalized products/components, perceive their different shapes, sizes, materials and orientation, and perform necessary system reconfiguration to achieve fast and flexible loading, unloading, clamping, etc.; (2) according to individual order requirements, accurately locate and dynamically organize production resources to achieve on-demand processing through the interaction and collaboration between manufacturing resources (equipment/products/workpieces). The realization of the above requirements largely depends on the application of DL in smart robots. For example, CNN is used to realize the identification and quality inspection of workpieces, and LSTM is used to realize short-term AGV flow prediction in the workshop [9]. Moreover, DL methods are adopted to identify machine faults and predict the remaining useful life (RUL) of such machinery as induction motors, gearboxes, and bearings [10,11].

(2) Human-centered collaboration paradigm

The concept of Industry 5.0 puts more emphasis on human-centered

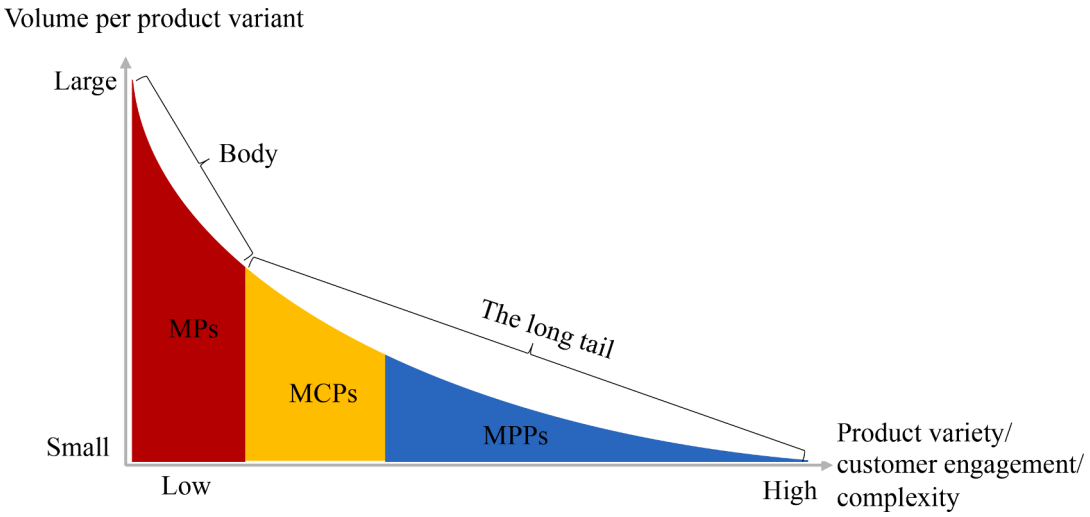


Fig. 1. The long tail production module.

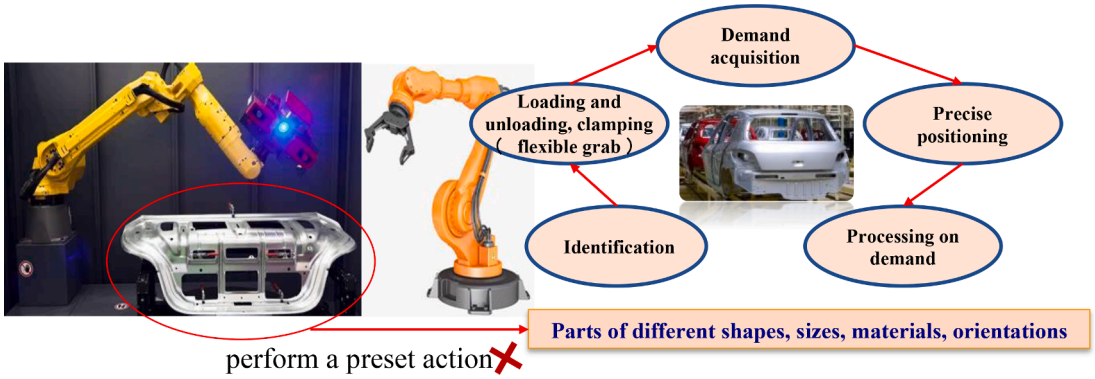


Fig. 2. The case of MPPs processing.

production and increases workers' job satisfaction and safety. Against this backdrop, in the process of mass personalized production, the efficient, intelligent and safe collaboration between humans and robots [12] can be realized through the interactive fusion of multimodal data, which interconnects and combines the strengths of humans and those of robots. In this area, DL has achieved relatively successful applications. For example, based on DL, it is now possible to realize the recognition of employee fatigue/unsafe operations, motion trajectory prediction, speech recognition, gesture recognition, emotion recognition, etc.

(3) Dynamic resource scheduling

MPPs are characterized by multi-variety, small-batch, and short-cycle production. These features of MPPs require the job-shop scheduling system to be more intelligent and responsive, so applying DL to production performance prediction and workshop resource scheduling can effectively increase the usefulness of scheduling to efficiently optimize resource management and usage for individual production needs and dynamic environments. In addition, the scheduling of computing and communication resource that supports intelligent decision-making and collaboration between manufacturing resources requires and uses DL to meet the requirements of deterministic routing, the right data arrival sequence and reliable computating. An example of this is the realization of short-term data traffic prediction and network resource allocation for AGVs based on DL.

2.3. Challenges of smart robots in mass personalization for deep learning

The development of MPPs calls for in-depth application of DL in smart industrial robots and puts forward higher requirements for DL's performance on the real-time response, optimization, adaptability, dynamism, multimodal data fusion and reliability.

(1) Real-time response: One major challenge of applying DL to industrial robots for MPPs comes from the uncertainties and dynamical conditions of the system configuration and production information. As the products are individualized and versatile, there are circumstances in which the planned operation has to be modified after the production process is initiated. In addition, MPPs may encounter urgent events that are disruptive to production, such as rush orders, machine failures, part-programming errors, and reconfiguration. To overcome such challenges, real-time decision-making capability needs to be incorporated into the manufacturing system [12]. This requires DL capable of making smart decisions quickly, usually in milliseconds.

(2) Optimization: MPPs have higher requirements for DL applications on the accuracy of prediction/classification and the optimization degree of resource scheduling for personalized modules. For example, the prerequisite for an effective predictive maintenance strategy is an accurate prediction of the RUL of the machine [13,14]. The results of the RUL prediction can guide companies to develop maintenance decision-making methods, implement remote maintenance to minimize unplanned machine downtime, and maximize service life to reduce maintenance costs. However, MPPs involve a wide variety of products, resulting in complex machine operating conditions, and the nonlinear decline of machine performance. Therefore, MPPs put forward higher accuracy requirements for DL-based predictive maintenance of machines.

(3) Adaptability: DL is required to be suitable for different processing scenarios of MPPs. For example, to ensure rapid response to consumer demands, the manufacturing system should have the flexibility in creating personalized modules with different features and assembling these modules with other modules from other manufacturers. This requires DL-based applications in smart industrial robots, such as parts recognition and quality inspection, to have adaptive learning capabilities and adapt to modules of all types - common, customized, or personalized.

(4) Dynamism: Facing MPPs, developing an optimal DLM is a considerable challenge. Note that a new DLM has to be used each time when the production of a new individualized product starts. Besides, various disrupting factors in the production workshop may occur and affect the practical activities of robots. These require the ability of smart robots to quickly and dynamically adjust and update the DLM network structure and parameters according to the changes in tasks.

(5) Multimodal data fusion: Multimodal data, such as images and videos, comes from different industrial robots and can provide more comprehensive and accurate information on workshop operation status through mutual support, supplement, and correction. By contrast, human-centered collaboration requires multimodal data fusion such as facial expressions, speech, and body movements to fully perceive, understand, and match the behavioral characteristics and habits of different workers. Traditionally, the feature fusion method is used for multimodal data fusion. However, due to the fact that multimodal data is collected at different locations and their features vary in size, meaning, redundancy, and importance for decision-making, how to achieve efficient multimodal data fusion is a key challenge for DL.

(6) Reliability: Most robotics applications are reliability-critical, such as material handling and human-robot collaboration. Unfortunately, DL-based methods are very fragile and susceptible to adversarial examples. An adversarial example which comes from erroneous sensor signals, exceptional data, etc. would cause the generation of unexpected or even false results when it is fed into DLMs. Since such adversarial perturbations can cause instability in DL performance, reliable training, deployment, and decision-making mechanisms of DLMs are essential. Therefore, mechanisms of reliable end-to-end communication and data verification should be designed to minimize the probability of data delay, data loss, and erroneous data for DLMs.

3. Cloud-edge-device collaboration based cloud manufacturing architecture

3.1. Overall framework

Although the central cloud for DL can effectively guarantee DL's optimization performance (by deploying and running large-scale DLMs), it usually suffers from latency and reliability issues because real-time decision-making cannot be guaranteed. Latency issues are caused by the bidirectional communication: between industrial robots and the cloud; between the cloud and actuators of locally-networked robots during the return of meaningful inference results from the cloud. Those latency issues may also lead to limited adaptability and dynamics of DL. The exceptional events that may occur during the bidirectional long-distance communication will harm the reliability of DL based decision-making applications in smart robotics. The long transmission delay of data also leads to reliability problems for time-sensitive DL applications for smart robotics.

This paper designed a hierarchical and distributed CMfg platform to address the aforementioned challenges. Specifically, through the deep integration of cyber-physical systems, edge computing, and CMfg, a multi-level and highly flexible CMfg system with "Cloud-Edge-Device" collaboration ability is built to support distributed training, deployment, operation, and update of DLMs, that satisfy MPPs' requirements for DL's performance on real-time response, optimization, adaptability, dynamism, multimodal data fusion and reliability. Then, flexible and efficient manufacturing of MPPs can be realized, quickly responding to a complex and changing buyer's market and meeting customers' needs for customized and personalized products. As shown in Fig. 3, the CMfg system framework with "Cloud-Edge-Device" collaboration is divided into three layers: the Device layer (production resource Layer), the Edge layer, and the Cloud layer.

(1) Device layer: The device layer mainly includes materials and smart industrial robots, such as AGVs, conveyors, robotic arms and other controlled platforms, which serve as "the physical layer" for the entire CMfg system. Different devices, machinery and equipment, such as assembly, welding, painting, packaging and machining robots, are controlled by their corresponding automatic control systems. Smart robots are equipped with more powerful sensors, such as 3D cameras, range and range sensors, to monitor the objects and the environments. DL can be implemented at the device layer in small and low-power embedded computing devices, such as FPGAs, DSP or ARM. Therefore, it is crucial to meet the real-time requirement for the device layer in MPPs' CMfg.

(2) Edge layer: The edge layer is generally composed of wireless base stations, small data centers (edge servers), and other edge nodes [15], which are usually deployed in the workshop/factory and mainly provide edge computing, storage, and networking services in the fixed or mobile way. The edge nodes are placed near the devices/robots, and can provide low-latency edge computing services for robot DL applications, but the computing power is relatively limited. For example, an edge node like an AI accelerator can be mounted on the robot and integrated into the robot controllers using hardware interfaces, permitting a simple and profitable combination of AI algorithms and robot control logic. The AI accelerator enables the efficient processing of neural network structures and the use of DL algorithms, such as for vision-based applications. Furthermore, the interconnection of physical devices (e.g., robots, machines and conveyor belts) and edge nodes using wires or wireless networks is implemented to compose manufacturing subsystems. Different manufacturing subsystems can meet different mission requirements, ensuring the adaptability and dynamics of DL.

(3) Cloud layer: In the cloud layer, a large number of servers and storage devices are used as the hardware base to support the mode of IaaS, SaaS, and PaaS, providing efficient parallel computing services, and large-scale data storage space for DL applications to ensure the optimization of DL-based decisions. The specific implementation can be a private cloud, developed by the large manufacturing enterprise and located inside its factory, or a third-party public cloud, built by the large IT vendor to provide cloud services for small and med-size enterprises. In either case, edge nodes would be placed much nearer to the robots (data sources) than that of the cloud.

Overall, in our designed architecture, edge nodes placed at the location as close as possible to the data source (smart robots), can deliver strong benefits, including faster insights, improved response times and better bandwidth availability. As smart robots' data is processed and analyzed closer to the point where the data is created, and the data does

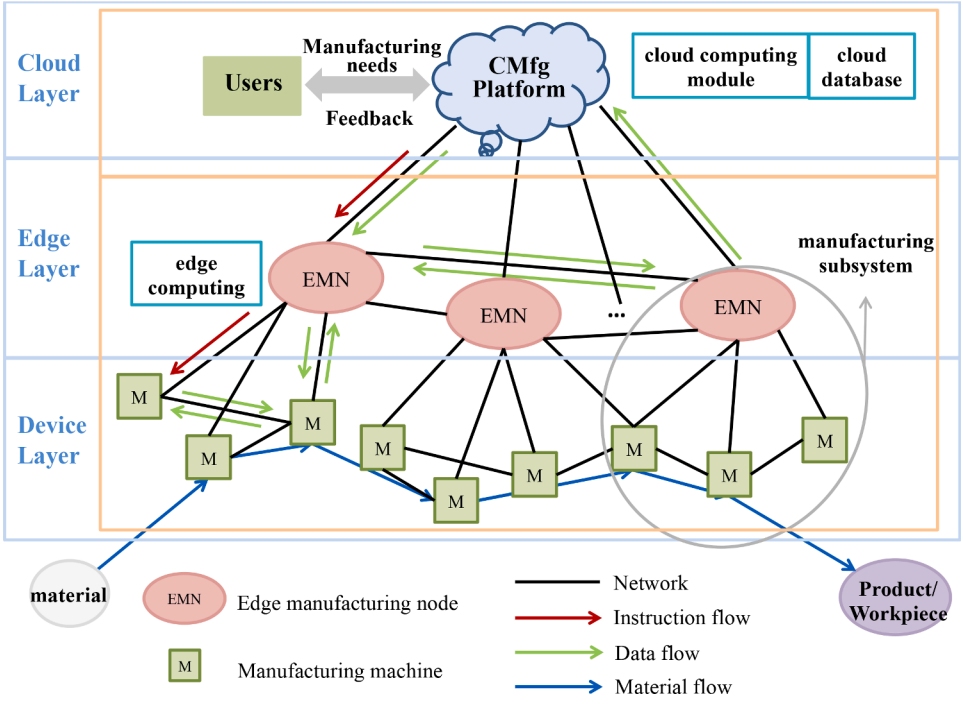


Fig. 3. Cloud-edge-device collaboration architecture.

not traverse over a network to a cloud to be processed, then latency and possible exceptional events (due to frequently unpredictable network bandwidth, latency or reliability) are significantly reduced. Meanwhile, the reliability of DL-based robotics applications is improved. In addition, reliable end-to-end communication (via multi-path transmission of multiple data copies) and data verification mechanism can be further introduced to minimize the probability of data delay, data loss, and erroneous data for DLMs.

3.2. Cloud-edge-device collaboration elements

(1) Data processing location: The data processing pipeline for DL generally consists of data collection, preprocessing, storage, analysis, and decision-making. Usually, the device layer takes the responsibility for data acquisition. The edge layer preprocesses the collected data to ease data analysis, then sends the resulting data to the cloud for storage and analysis, and issues decisions to control the devices. However, in the cloud-edge-device framework, each stage of a data processing pipeline may occur at the device, edge, or cloud layers, depending on the design of smart decision-making mechanisms. For example, the device layer may be responsible not only for data collection but also for data preprocessing and simple decision-making; the edge layer may also undertake the data preprocessing and real-time decision-making functions.

(2) Offline/Online learning: The data processing and analytics for DL can be performed in an online or offline way. Offline training, decision-making and updating of large-scale DLMs using massive multi-dimensional and multi-source data are usually performed in the cloud with abundant computing and storage resources for time-insensitive applications. For edge nodes and smart robots near to data sources or shopfloor *things*, they are normally used to conduct online incremental training and updating of lightweight DLMs based on new local data for time-critical applications. They can also collaboratively realize online training, decision-making and updating of large-scale DLMs through joint training and collaborative approaches.

(3) Data sharing: There are three kinds of data-sharing among the cloud, edges, and devices: data, information, and knowledge. Data is collected from IoT sensors of smart robots. Information is useful data, obtained through data preprocessing methods, including data cleaning, transformation, extraction, merging and so on. Knowledge is the operating principle or law of the system obtained via information processing. Data or information sharing is common, as end devices and edge nodes can perform data collection or preprocessing. In the cloud-edge-device collaboration architecture, knowledge sharing can contribute to the reduction of data transmission, e.g., extracted features and DLM parameters can be shared among the cloud, edge, and device.

(4) The life cycle of a DLM: It usually includes the stages of design, training, operation and update. The entire life cycle of a DLM is normally performed on high performance servers with massive data. However, in the framework of cloud-edge-device collaboration, activities at each stage of the DL life cycle may occur at the device, edge, or cloud layers. For example, a lightweight DLM obtained through parameter sharing and pruning requires less computing resource while the accuracy of DLM inference can be ensured. This makes it possible to train and update DLMs on the edge nodes and end devices.

4. Deployment and update mechanism of decision-making models

4.1. Mechanism

4.1.1. Vertically distributed deployment of DLMs

(1) Conventional Framework: The Cloud centers with abundant computing and storage resources are responsible for storing/processing massive IIoT (Industrial IoT) data and creating/managing DLMs for various edge nodes and end robots. The Edge/Device layer is designed to flexibly utilize the result from DLM and interact with industrial robotics. Specifically, the Edge nodes/Devices collect various sensors data, which has different geographic-coverage ranges and formats, and upload the data to the cloud for intelligent analytics. In addition, the frequency and methods of data collection are also different. Therefore, it is essential to integrate data in a suitable format and preprocess it for DL. The cloud uses the cleaned data to create multiple DLMs, according to the characteristics of Edges/Devices and IIoT data. Then it chooses the best DLM model that fits the individual task, and sends the model to the Edge/Device along with the usage information.

As for updating Edge/Device DLMs to improve model performance, centralized or local update strategies (offline/online) can be used: (1) As shown in Fig. 4(a), the centralized update means that the cloud uses retraining, continuous training, or incremental learning methods to update the existing DLMs with new data from the Edge/Device, and then redeploys the updated model to the Edge/Device. Note that retraining large and complex DLMs from scratch using all the available training data will take much time and computing resources, while continuous training may lead to overfitting. Typically, all parameters of a DLM should be updated. However, as new data collected from practical applications have similar patterns to the existing dataset, incremental learning based on local parameter update is usually used to improve the DLM's accuracy in a short time and avoid overfitting [16]. (2) as shown in Fig. 4(b), the local update means that the DLMs deployed on the Edge nodes/Devices are separately further trained, and updated using native data to utilize the local data better and meet the needs of personalized tasks.

Overall, in this deployment and update mode of the DLMs, the device layer and edge layer are not only responsible for the collection and preprocessing of raw data, but also for decision-making, which can avoid the delay problem that may exist in cloud-edge communication, and effectively guarantee the real-time performance of DLMs decision-making for smart industrial robots. At the same time, this mode allows DLMs to be trained and updated according to personalized tasks, with certain dynamics and adaptability. However, there are some unavoidable problems in this mode, (1) Due to the limited resource of the Edge/Device, it is impossible to run large-scale DLMs, which may lead to poor optimization performance. Therefore, it is necessary to design lightweight DLMs. (2) In a centralized update mechanism, the preprocessed data needs to be uploaded to the cloud for model training. If the network is congested (leading to a long delay), it may cause the problem of late-model updates. 3) In the local update mechanism, the small amount of local data may lead to the problem of DLM overfitting.

(2) Distributed Deep Learning (DDL) Framework: Large DLMs are deployed in the cloud, while Edge nodes/end devices can only support shallow/lightweight DLMs. Therefore, a hierarchical exit mechanism for collaborative decision-making [17] can be adopted, as shown in Fig. 5, by performing a portion of the DLM inference computation on the edge rather than sending the raw input to the cloud. Using an exit point after Edge inference, we may decide on those samples that the Edge-DLM is confident about without sending any data to the Cloud. For more complicated cases, the output of the Edge-DLM as intermediate results is sent to the cloud, where further inference is performed using additional neural network layers to achieve higher accuracy, and a final decision is made. As shown in (d), this framework can be extended to multiple edge nodes that may be geographically distributed. Here, each edge node performs local computation as in (c). If the local exit point is not confident about the sample, each edge node sends intermediate output (a feature of raw data) to the cloud, where feature aggregation is performed before making a final decision. Note that each feature aggregation or fusion method makes different assumptions about how the edge output should be combined, resulting in different system accuracy. Aggregation methods can be: (1) Max pooling, which aggregates the input vectors by taking the max of each component. (2) Average pooling, which aggregates the input vectors by taking the average of each component. (3) Series connection, which simply concatenates the input vectors together (Fig. 6).

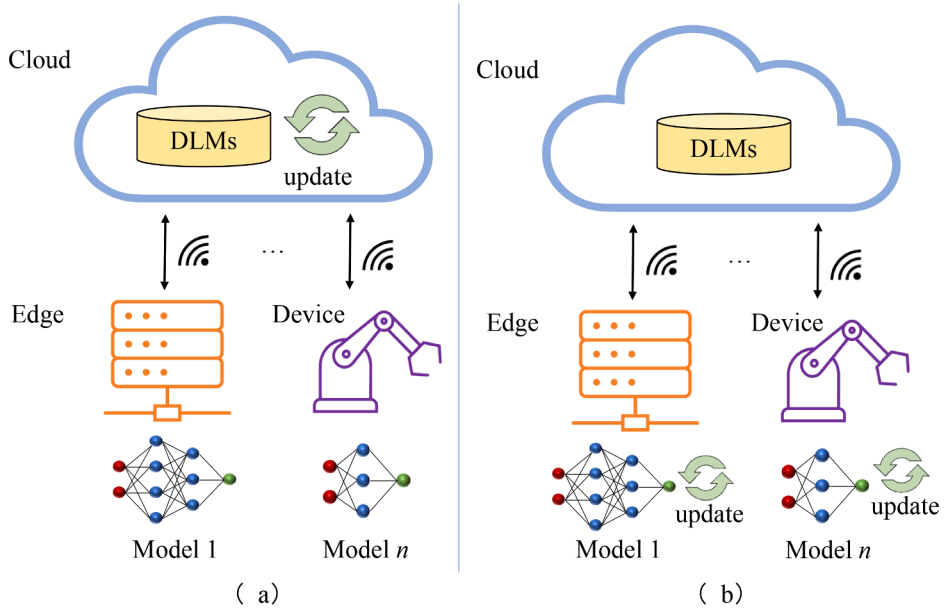


Fig. 4. The update mechanism of Edge/Device DLM: (a) centralized update mechanism, (b) local update mechanism.

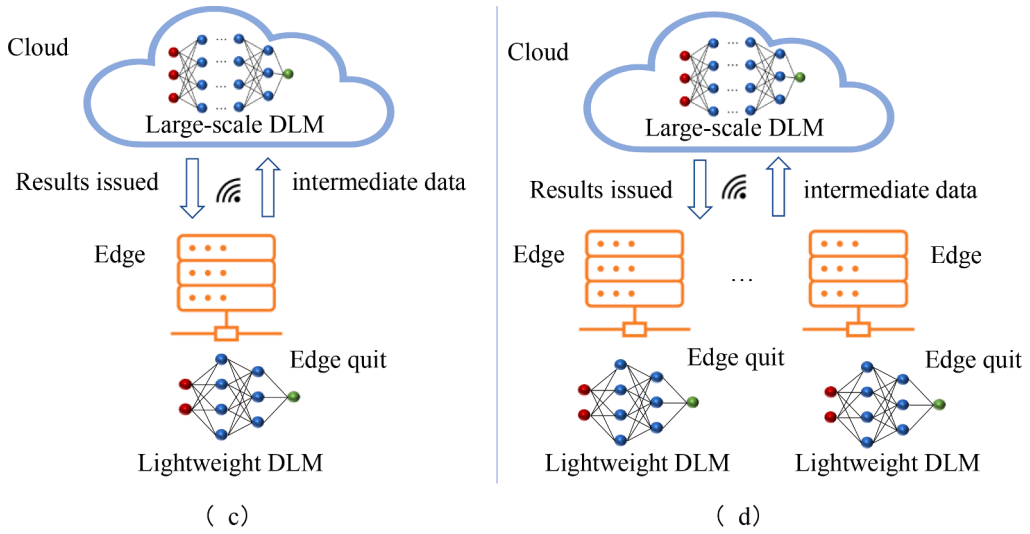


Fig. 5. Cloud-edge deployment framework for DLMs. (c) introduces an edge node and a local exit point that may make decisions on samples before the cloud, (d) extends (c) by adding more edge nodes, the results are aggregated for decision-making.

The vertical expansion can be achieved by adding an end device, as shown in (e). The edge node functions like the cloud, obtaining output from the end device, making a decision if possible, and forwarding its output to the cloud for further processing till a final decision is reached. In this way, samples that can be used to make correct decisions locally exist without any communication with the edge or cloud. Otherwise, the features extracted from samples are sent to the edge, and eventually to the cloud if necessary. This deployment framework can also be scaled geographically across the edge or device layers, as shown in (f) and (g).

As for updating DLMs to improve model performance, centralized joint update or local update (offline/online) can be applied: (1) Centralized joint update means that the DDL model can be trained on a single powerful server or in the cloud, and then deployed to the edges/devices. During training, the loss from each exit is combined during backpropagation so that the entire DDL model can be jointly trained, and each exit point achieves good accuracy related to its depth. (2) local update means that the DL models deployed on the device/edge/cloud are separately trained and updated according to their collaborative mechanisms, using native data to utilize the local data better.

Overall, this deployment and update mode of DLMs has the following advantages: (1) DLMs of different scales are deployed on the cloud-edge-device and coordinated through the layered exit mechanism, which has the potential to satisfy the requirements of smart industrial robots for optimal real-time decision-making. (2) The intermediate Edge-DLM output (feature extraction) can be designed to be much smaller than the raw data (e.g., a raw image from a camera); therefore, communication costs required between the Edge and the Cloud can be drastically reduced. (3) It allows DLMs to be trained and updated according to personalized tasks, which can effectively meet the dynamic and adaptive requirements of MPPs for DL. However, there are still some inevitable problems with this model. For example, in a local update mechanism, an edge/device collects only a small amount of data within a limited area, and training a DLM from scratch can cause over-fitting problems and consume many computing resources [18]. A common way to solve this problem is using the Cloud-DLM to help train the Edge/Device-DLM [19]. When the core network is underloaded, the edge/device uploads

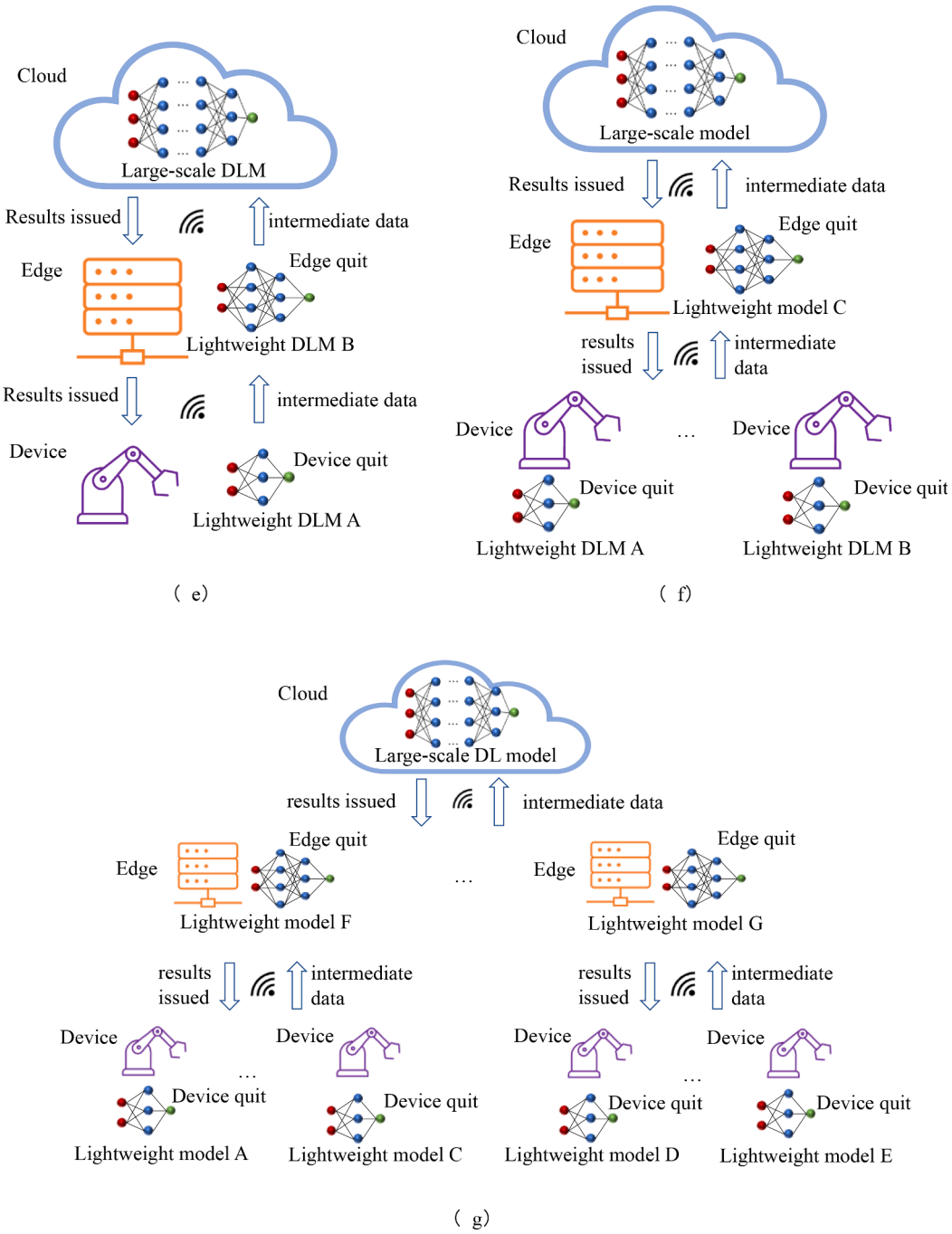


Fig. 6. Cloud-edge-device deployment framework for DLMs. (e) extends (c) by adding an end device, (f) extends (e) by adding multiple end devices results of which are aggregated for decision making, and (g) shows how the edges can also be distributed like the end devices.

the raw data to the cloud. Based on these data, the cloud uses the Cloud-DLM to predict the labels and sends them back to the edge/-device. Finally, the labelled samples are used to retrain the Edge/Device-DLM. This process is ongoing and adaptive. As the number of labelled samples increases, the Edge/Device-DLM can further improve its accuracy. Furthermore, using the deep convolutional neural network to share its m (where m is a positive integer) lower layers to assist in training the shallow network is also an effective approach. The Cloud-DLM first sends its m lower layers to the edge/device, where some convolutional layers, some fully connected layers, and one softmax layer are connected to m lower layers to form the shallow DLM. Then, the shallow DLM is trained by freezing its m lower layers and fine-tuning the remaining layers with small data. This means that the shallow DLM uses the knowledge of the Cloud-DLM to assist its training.

(3) Segmented deep learning (SDL) framework: The DLM will be divided and deployed on the cloud-edge device. The cloud, edge, and devices will be responsible for the computing tasks of different segmented models to generate final decision results. As shown in Fig. 7, the cases include cloud-edge-device joint deployment, cloud-edge joint deployment, and edge-device joint deployment based on different computing subjects participating in computing. At the same time, by means of the location of data and decision-making, the decision-making process can be divided into two ways: upward and downward decision-making. For example, as shown in (h), in the cloud-edge-device joint deployment mode for bearing fault diagnosis, the upward decision indicates that the end device is not only used to collect vibration data for

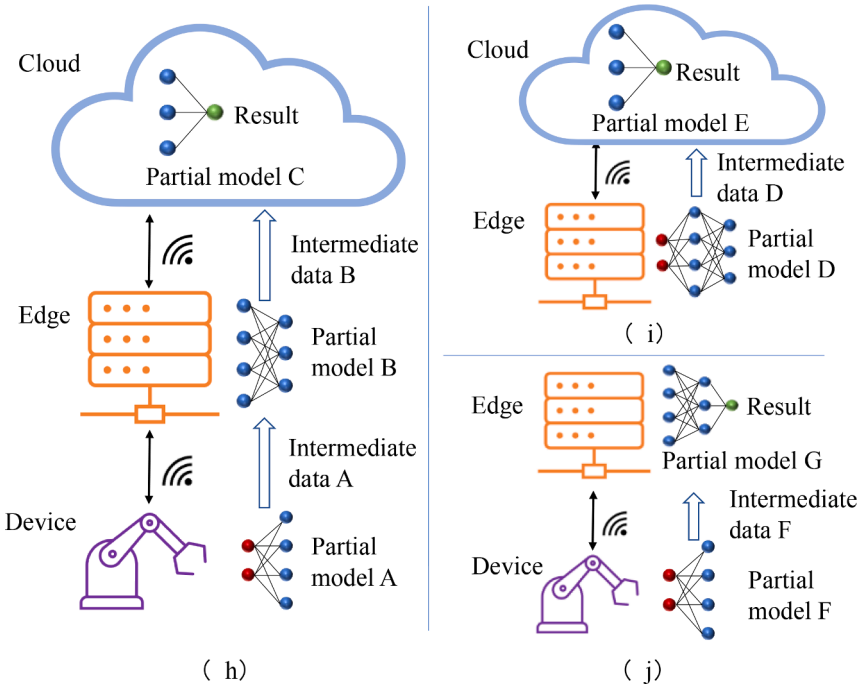


Fig. 7. SDL Framework. (h) shows cloud-edge device joint deployment, (i) shows cloud-edge joint deployment, (j) shows edge-device joint deployment.

fault diagnosis, but also undertakes part of the computing task of the DLM; then it transmits the intermediate data to the edge. The edge continues to perform the decision-making work, then transmits the processing results of the edge to the cloud, and the cloud continues to perform the decision-making work to obtain the fault type. In downward decision-making, new data is input from the cloud, and the result is obtained on the end device.

As for updating SDL models to improve model performance, centralized training and update can be adopted. Fig. 8 shows that the entire SDL model can be trained on a single powerful server or in the cloud, then split and distributed to the cloud/edges/devices. In terms of training methods, retraining, continuous training, and incremental learning can be adopted under different application requirements.

Overall, this deployment and update mode of DLMs has the following advantages: (1) Cloud, edge, and device jointly serve a large-scale DLM, which can ensure optimal decision-making. (2) The original data is replaced by the intermediate feature data, which can effectively reduce the cloud-edge-device communication traffic, and provide better privacy protection. However, this model still has some inevitable problems: (1) Transmission of the intermediate data between partitioned DLM parts is inevitable. This transmission will surely bring more delay for end-to-end deep learning applications. Also, the size of intermediate data is proportional to the size of the neural network layer where the division of the DLM occurs. If a DLM has been divided at a layer containing 1000 neurons, the intermediate data size will be 1000 times the output parameter size [20]. Therefore, this joint deployment method

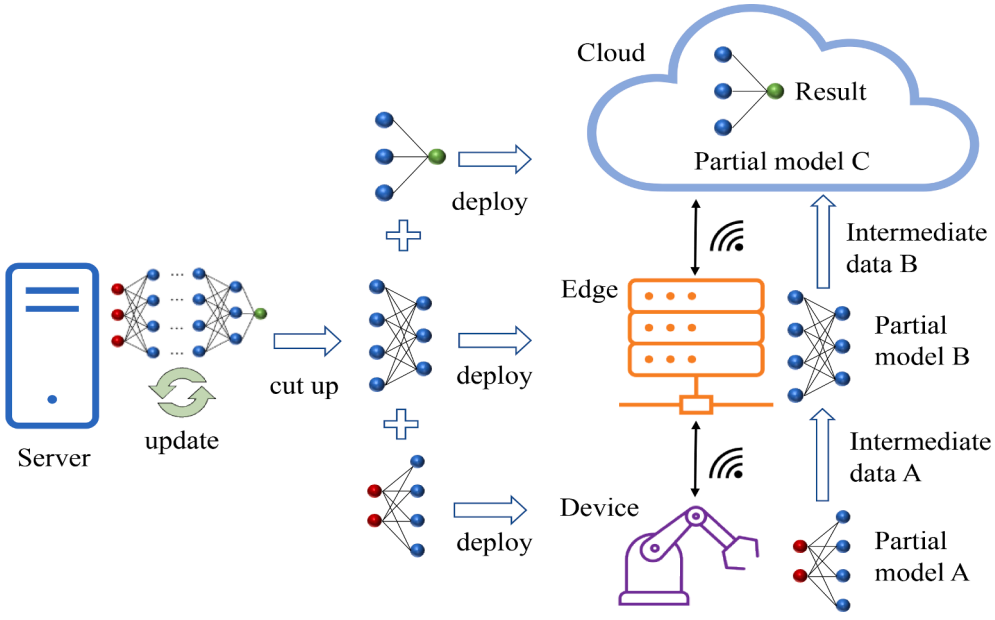


Fig. 8. Centralized training and updated mechanism of SDL.

focuses on finding a suitable model split point and achieving a reasonable balance between the amount of computation and the amount of communication. (2) Each DLM update needs to go through the costly process of training, segmentation, deployment, etc., which decreases the adaptability and dynamism of DLMs to personalized tasks.

Potential application field: In the fault diagnosis [21] of robot machining tools, a lightweight DLM based on single-source information, such as vibration/current signals, on the edge node/end device can be deployed to realize real-time fault diagnosis. If the result does not achieve the acceptable accuracy/confidence, uploading extracted intermediate data (such as feature vectors) to the cloud and using a large-scale DLM for multi-source information fusion can ensure the reliability of decision-making for fault diagnosis.

4.1.2. Horizontally distributed deployment of DLMs

(1) Horizontal segmented deep learning (HSDL) framework: As shown in Fig. 9, according to the different computing power of cloud/edges/devices, the model is split and allocated to different computing nodes so that each computing node executes a part of the model to jointly make decisions, which can reduce the computing pressure of a single node.

As for updating HSDL models for better model performance, centralized training and update can be adopted. It means that the cloud center or a single powerful server adopts suitable training methods such as incremental learning, retraining, continuous training to train the model based on the new data, and then splits the trained model and distributes it to the cloud/edges/devices.

This mode is similar to SDL, which inherits the advantages and disadvantages of SDL. The difference is that the data of HSDL flows in the same layer, while SDL flows across layers.

(2) Federated learning framework: Federated learning (FL) is a distributed machine learning strategy that generates a global model by learning from multiple decentralized edges/devices. Specifically, a virtual shared model is established on a public node, other nodes update parameters to the public node without violating privacy regulations, and the virtual model aggregates all the parameter data to form an optimal model. FL enables on-device training, keeps the client's local data private, and updates the global model based on the local model updates. While FL methods offer several advantages, including scalability and data privacy, they assume there are available computing resources at each edge/device. However, the IIoT-enabled devices, e.g., robots, CNC, and low-cost computing devices, may have limited processing ability, low bandwidth and power, or limited storage capacity. How to train distributed machine learning models for resource-constrained IIoT devices is still an open question.

Potential Application Field: In smart grasping and manipulation, computer vision can help the robot figure out how to grasp the items of different types. Deep learning is applied to improve the robot grasps over time. Data of a robot is shared with other robots through the cloud. In such a scene, FL can be adopted to generate a global grasping DLM by learning from multiple decentralized edges/robots. The above approach can effectively prevent the leakage of private information while completing the sharing of previous grasping experience/knowledge of the robots.

4.1.3. Data Preparation for DLMs

Another important dimension for DLMs is data preparation. For DL, there is a famous saying "garbage in, garbage out", so preparing good data can help significantly improve the model performance. First, the engineers should understand the application scenario of smart robots for MPPs and find out the main influencing factors of decision-making. Second, after the raw data about the influencing factors is obtained, it should be cleaned first to improve the quality of data. Such a difficult but important process may involve dealing with incomplete, inconsistent, exceptional, noisy, erroneous data [22]. Third, the well-prepared high-quality data can then be used for DLMs. Following the mechanisms introduced in Sections 4.1.1 and 4.1.2, those DLMs can be trained, deployed and updated based on those data. Four, the effect of DLMs for smart robot decision-making can be evaluated according to the outcomes. Five, the results can be used to improve the data acquisitions, data preparation and collaborative mechanisms of DLMs until the whole system can work stably and produce satisfactory results. The data cleaning approaches should also be adopted in the robots (devices), edge nodes and cloud centers [23], when those nodes are responsible for data preparation and model training & updating.

4.2. Key technologies

Incremental learning: Incremental learning can make full use of newly generated data, and its advantages are mainly manifested in two aspects: on the one hand, because it does not need to save historical data, it reduces the storage space occupation; On the other hand, the incremental learning of the new samples makes full use of the historical training results, thereby significantly reducing the time of subsequent training.

Transfer learning: Transfer learning is a new machine learning method that uses existing knowledge to solve problems in different but related fields. It relaxes two basic assumptions in traditional machine learning: (1) The training samples used for learning and the new test samples satisfy the condition of independent and identical distribution; (2) There must be enough training samples available to learn a good intelligence model. In general, the purpose of transfer learning is to transfer existing knowledge to solve learning problems with little or no labelled sample data in the target domain.

DLM compression: The computing resources at the edge/device are relatively limited. For effective training of DLMs at the edge/device, it is

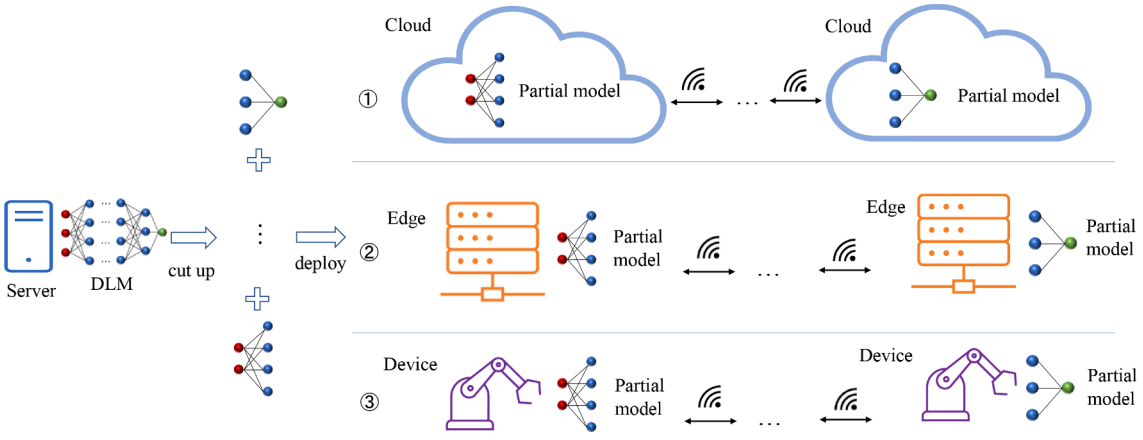


Fig. 9. HSDL Framework.

important to reduce their computing resource requirement. Therefore, the compression of the model and the design of lightweight models are the key technologies. For example, in the case of DLMs, insensitive parameters can be reduced through parameter sharing and pruning, and then storage, communication, and computational overhead can be reduced.

DLM segmentation: The model needs to be split into SDL or HSDL systems. Therefore, it is necessary to find an appropriate cutting point, try to keep the computationally complex work in the cloud center, and cut DLM in the place with the least amount of communication so as to realize the trade-off between the amount of calculation, and the amount of communication [24].

DLM selection technology: There is a lax trend for DLMs: the larger the model size (the higher the number of parameters), the higher the accuracy. However, in edge/device computing scenarios, on the one hand, resource constraints cannot afford large-scale DLMs; on the other hand, different scenarios have different requirements for accuracy. In many cases, part of the accuracy can be sacrificed in exchange for better real-time performance. The DLM selection technology aims to seek a better trade-off between the resource consumption and accuracy of the model, and obtain the model that best meets the needs of the scenario.

5. Conclusions and future work

In this paper, a cloud-edge-device collaboration CMfg framework is proposed to address the high requirements for deep learning models (DLMs) of smart robots in mass personalization. Under this framework, different deployment and update mechanisms of DLMs are discussed in detail to support rapid response and high-performance decision-making of smart industrial robots for MPPs. In addition, key technologies such as incremental learning, transfer learning, DLM compression, DLM segmentation, and Model selection technology are discussed and can provide references for research directions in this field. Our future work includes designing joint training and updating algorithms for collaborative DLMs on cloud-edge-device, exploring theories and methods for the establishment of lightweight DLMs, and implementing typical industrial applications.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB1715700, in part by the National Natural Science Foundation of China under Grant 62103046 and 72192844, in part by the Fundamental Research Funds for the Central Universities under Grant E1E40805X2 and the State Key Laboratory of Digital Manufacturing Equipment and Technology under Grant DMETKF2021012.

References

[1] T.Y. Lin, et al., Efficient container virtualization-based digital twin simulation of smart industrial systems, J. Clean. Prod. 281 (2021).
[2] F. Tao, J. Cheng, Q. Qi, IIHub: an industrial Internet-of-Things hub toward smart manufacturing based on cyber-physical system, IEEE Trans. Ind. Inform. 14 (5) (2017) 2271–2280.
[3] C. Yang, S. Lan, L. Wang, W. Shen, G.Q. Huang, Big data driven edge-cloud collaboration architecture for cloud manufacturing: a software defined perspective, IEEE Access 8 (2020) 45938–45950.
[4] C. Yang, F. Liao, S. Lan, L. Wang, W. Shen, G.Q. Huang, Flexible resource scheduling for software-defined cloud manufacturing with edge computing, Engineering (2021).
[5] X. Li, P. Zheng, J. Bao, L. Gao, X. Xu, Achieving cognitive mass personalization via the self-X cognitive manufacturing network: an industrial-knowledge-graph-and graph-embedding-enabled pathway, Engineering (2021).
[6] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.
[7] C. Yang, W. Shen, X. Wang, The internet of things in manufacturing: key issues and potential applications, IEEE Syst. Man Cybern. Mag. 4 (1) (2018) 6–15.
[8] S.J. Hu, Evolving paradigms of manufacturing: from mass production to mass customization and personalization, Procedia CIRP 7 (2013) 3–8.
[9] Z. Zhao, W. Chen, X. Wu, P.C.Y. Chen, J. Liu, LSTM network: a deep learning approach for short-term traffic forecast, IET Intell. Transp. Syst. 11 (2) (2017) 68–75.
[10] L. Zuo, L. Zhang, Z.H. Zhang, X.L. Luo, Y. Liu, A spiking neural network-based approach to bearing fault diagnosis, J. Manuf. Syst. 61 (2021) 714–724.
[11] Y. Ding, M. Jia, Q. Miao, P. Huang, Remaining useful life estimation using deep metric transfer learning for kernel regression, Reliab. Eng. Syst. Saf. 212 (2021), 107583.
[12] H. Zhong, X. Li, L. Gao, C. Li, Toward safe human-robot interaction: a fast-response admittance control method for series elastic actuator, IEEE Trans. Autom. Sci. Eng. (2021).
[13] J. Wan, et al., A manufacturing big data solution for active preventive maintenance, IEEE Trans. Ind. Inform. 13 (4) (2017) 2039–2047.
[14] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, J. Lin, Machinery health prognostics: a systematic review from data acquisition to RUL prediction, Mech. Syst. Signal Process. 104 (2018) 799–834.
[15] H. Yamanaka, E. Kawai, Y. Teranishi, H. Harai, Proximity-aware IaaS in an edge computing environment with user dynamics, IEEE Trans. Netw. Serv. Manag. 16 (3) (2019) 1282–1296.
[16] L. Ren, Y. Liu, X. Wang, J. Lu, M.J. Deen, Cloud-edge-based lightweight temporal convolutional networks for remaining useful life prediction in IIoT, IEEE Internet Things J. 8 (16) (2021) 12578–12587.
[17] S. Teerapittayanon, B. McDanel, H.T. Kung, Distributed deep neural networks over the cloud, the edge and end devices, in: Proceedings of the International Conference on Distributed Computing Systems, 2017, pp. 328–339.
[18] C. Ding, A. Zhou, Y. Liu, R. Chang, C.H. Hsu, S. Wang, A cloud-edge collaboration framework for cognitive service, IEEE Trans. Cloud Comput. (2020).
[19] T. Jing, X. Tian, H. Hu, L. Ma, Cloud-edge collaboration framework with deep learning-based for remaining useful life prediction of machinery, IEEE Trans. Ind. Inform. (2021).
[20] S. Kum, Y. Kim, J. Moon, Deploying deep neural network on edge-cloud environment, in: Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), 2019, pp. 242–244.
[21] L. Wen, X. Li, L. Gao, Y. Zhang, A new convolutional neural network-based data-driven fault diagnosis method, IEEE Trans. Ind. Electron. 65 (7) (2017) 5990–5998.
[22] X. Wang, C. Wang, Time series data cleaning: a survey, IEEE Access 8 (2019) 1866–1881.
[23] C. Wang, X. Huang, J. Qiao, T. Jiang, L. Rui, J. Zhang, J. Sun, Apache IoTDB: time-series database for internet of things, Proc. VLDB Endow. 13 (12) (2020) 2901–2904.
[24] Y. Kang, et al., Neurosurgeon: collaborative intelligence between the cloud and mobile edge, ACM SIGARCH Comput. Archit. News 45 (1) (2017) 615–629.