12-2021

Slicing-Based Artificial Intelligence Service Provisioning on the Network Edge: Balancing AI Service Performance and Resource Consumption of Data Management

Mushu Li

Jie Gao

Conghao Zhou

Xuemin Shen

Weihua Zhuang

IEEE Vehicular Technology Magazine, Vol. 16, No. 4 (December 2021): 16-26. DOI.

Slicing-Based Artificial Intelligence Service Provisioning on the Network Edge: Balancing AI Service Performance and Resource Consumption of Data Management

Mushu Li
University of Waterloo, Ontario, Canada
Jie Gao
Marquette University, Milwaukee, Wisconsin
Conghao Zhou
University of Waterloo, Ontario, Canada
Xuemin Sherman Shen
University of Waterloo, Ontario, Canada
Weihua Zhuang
University of Waterloo, Ontario, Canada

Abstract:

Edge intelligence leverages computing resources on the network edge to provide artificial intelligence (AI) services close to network users. As it enables fast inference and distributed learning, edge intelligence is envisioned to be an important component of 6G networks. In this article, we investigate AI service provisioning for supporting edge intelligence. First, we present the features and requirements of AI services. Then we introduce AI service data management and customize network slicing for AI services. Specifically, we propose a novel resource-pooling method to regularize service data exchange within the network edge while allocating network resources for AI services. Using this method, network resources can be properly allocated to network slices to fulfill AI service requirements. A trace-driven case study demonstrates that the proposed method can allow network slicing to satisfy diverse AI service performance requirements via the flexible selection of resource-pooling policies. In this study, we illustrate the necessity, challenge, and potential of AI service provisioning on the network edge and provide insights into resource management for AI services.

Edge intelligence leverages computing resources on the network edge to provide artificial intelligence (AI) services close to network users. As it enables fast inference and distributed learning, edge intelligence is envisioned to be an important component of 6G networks. In this article, we investigate AI service provisioning for supporting edge intelligence. First, we present the features and requirements of AI services. Then we introduce AI service data management and customize network slicing for AI services. Specifically, we propose a novel resource-pooling method to regularize service data exchange within the network edge while allocating network resources for AI services. Using this method, network resources can be properly allocated to network slices to fulfill AI service requirements. A trace-driven case study demonstrates that the proposed method can allow network slicing to satisfy diverse AI service performance requirements via the flexible selection of resource-pooling policies. In this study, we illustrate the necessity, challenge, and potential of AI service provisioning on the network edge and provide insights into resource management for AI services.

Introduction

6G networks are envisioned to support many emerging use cases, such as extended reality, remote health care, and autonomous systems [1], [2]. Compared with services supported by 5G networks, services in the 6G era will be even more diverse, potentially blurring the boundaries among enhanced mobile broadband, massive

machine-type communications, and ultrareliable and low-latency communications. Such services will demand highly intelligent and flexible networks, driving a confluence of advanced networking and AI technologies.

AI can play an essential role in network management, e.g., resource management [3], [4] and protocol design [5]. Meanwhile, with recent advancements in machine learning algorithms, many network services have integrated AI techniques into applications, such as object detection in autonomous vehicles, and learning-based language processing. These services are referred to as *AI services*. Because AI services must gather or generate a vast amount of data, edge intelligence has attracted extensive interest as it moves AI closer to user devices (UDs) and alleviates data traffic load in the core network. Empowered by distributed learning techniques, edge intelligence leverages the communication, computing, and storage resources at each edge node, i.e., a base station (BS) or other access points (APs), to process data.

Typically, an AI service involves two phases: inference and model training. Different from conventional services, AI services largely depend on the data generated by UDs, and such dependence exists in both phases. For example, image-recognition services depend on images and their corresponding labels uploaded from UDs. As a result, the availability and quantity of data from UDs determine the effectiveness of an AI service, including inference accuracy and learning speed. For example, inference accuracy may increase when more data are available at an edge node. At an edge node, specifically, there are two types of data available for AI services: data collected from UDs by this edge node and data shared by other edge nodes. Although data sharing among all edge nodes increases the amount of available data for an edge node and potentially improves the performance of AI services, it can consume significant network resources. In particular, each edge node needs excessive computing resources for data processing and communication resources for exchanging data with other edge nodes. Considering that the amount of data collected by each edge node can be very different, a viable alternative to sharing all data is to migrate a portion of the data from the edge nodes that have collected sufficient data to those that need more. Achieving this requires scalable and on-demand network resource management, especially considering that AI services need to coexist and share resources with conventional services. Although a few existing works, such as [6] and [7], have studied the relationship between the performance of AI services and network resource allocation, the topic needs further investigation.

As a major innovation in 5G technology, network slicing can support a multitude of network services with diverse service requirements by creating and maintaining logically isolated virtual networks, i.e., slices, for different services [1]. Network slicing has the potential to support AI services in future networks; however, due to the unique features and requirements of AI services, a slicing-based network should not treat them in the same way that it does conventional services. The reason is two-fold. First, AI services have unique performance metrics, such as accuracy, which require the coordination of data available to edge nodes, while network slicing considers conventional performance metrics, such as throughput and delay. Second, the location of physical resources can impact the performance of AI services, which complicates resource management and network operation in network slicing.

In the following sections, we investigate AI service provisioning on the network edge and extend network slicing to support AI services. Specifically, we propose a resource-pooling method, which customizes resource virtualization for each AI service by considering the location of physical resources and enabling effectual data migration among edge nodes. The proposed resource-pooling method addresses the aforementioned challenges in the existing network slicing framework. Furthermore, we provide a case study to demonstrate the effectiveness of the resource-pooling method for AI services.

AI Services and Requirements

AI Services on the Network Edge

Similar to conventional services, an AI service is enabled by a chain of service functions. The difference is that, in an AI service, one or more functions are based on AI models, such as deep neural networks (DNNs) and *k*-nearest neighbor algorithms. We refer to these functions as *AI functions*. In existing networks, AI functions are deployed mostly in a cloud server, while edge nodes simply forward the data of UDs to the cloud server. The disadvantage of such cloud-centric AI service provisioning is a heavy data traffic load on the core network. To address this issue, some AI functions can be deployed at the network edge to be close to UDs. In such a case, edge nodes can play an active role to support AI services in the following scenarios:

- *Edge-assisted, cloud-hosted AI scenario*: A small portion of AI functions, such as data preprocessing and aggregation, are deployed at the network edge, while the remaining AI functions are executed at the cloud server.
- *Cloud-assisted, edge-hosted AI scenario*: All the AI functions are placed at the network edge for inference, and the cloud server assists the network edge in training the AI models used by AI functions. The cloud server coordinates data exchange among edge nodes. An example is presented as "AI Service 1" in Figure 1.
- *Fully edge-hosted AI scenario*: All the AI functions are deployed at the network edge, and the edge nodes exchange information with each other for training AI models. An example is shown as "AI Service 2" in Figure 1.

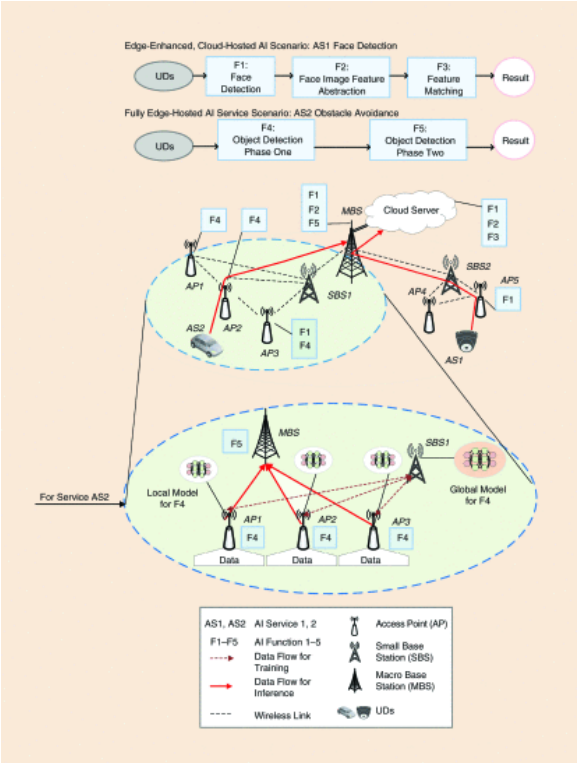

Figure 1 Service management for AI services on the network edge.

A comparison of the aforementioned three scenarios is summarized in Table 1.

Table 1 The three scenarios of AI services in edge intelligence.

	Edge-Assist-Hosted AI Scenario	Cloud-Assisted, Edge-Hosted AI Scenario	Fully Edge-Hosted AI Scenario
Use cases	Image and voice recognition	Automated driving, mobile virtual reality	Business informatics, smart city
Key resource in demand	Communication	Computing	Computing
Role of edge	Data preprocessing and aggregation	Inference	Inference and model training
Requirements and features of the service	Large data size, which requires a database in the cloud server	Stringent service requirement, which requires real-time training and fast inference	Demand for fast inference and privacy-preserving measures
Learning methods	Centralized learning	Federated learning, splitting learning, and so on	Transfer learning, gossip learning, and so forth

Key Performance Indicators

Because AI services can be viewed as a special type of compute-intensive service, conventional performance indicators such as latency and energy efficiency apply to them. In addition, the following new performance indicators are necessary for evaluating the performance of AI services:

- Accuracy [6], [7], measuring the difference between the inference results derived by an AI service and real values.
- Learning speed [7], [8], measuring how fast an AI model can be fully trained. For example, for DNNs, the learning speed is the convergence rate of the loss function during the training process.

Moreover, other performance indicators, such as running time [9] and memory shrinks [10], can also be applied for evaluating the performance of an AI service.

Features of AI Services

As mentioned previously, in general, an AI service consists of two phases: inference and model training. In inference, edge nodes process the data collected from UDs and deliver computing results to UDs, which is similar to conventional computing services. For model training, the data available to an edge node includes the data collected from UDs and the data migrated from other edge nodes. Each edge node utilizes its available data to train the AI models used by AI functions and exchanges training parameters with other nodes to improve the effectiveness of training. For example, in federated learning, edge nodes train their local AI models, upload the parameters of local models to a centralized node, and obtain the parameters of a global model from the centralized node periodically.

For both inference and model training, data flows from UDs to edge nodes as well as among edge nodes are necessary. In the inference phase, the way that data flow among edge nodes affects the performance, e.g., inference delay, of AI functions. In the training phase, the way that data flow among edge nodes affects the performance of AI functions from the following three aspects. First, the migration of data among edge nodes determines how model training is performed. We define a term, i.e., *learning structure*, to specify which edge nodes train the AI model of an AI function and how they migrate data with each other in the network. If data from UDs are migrated to fewer edge nodes for training, the learning structure is more centralized, and the benefit is a higher learning speed and inference accuracy. Second, the migration of data among edge nodes balances the available data at the edge nodes and alleviates data bias. This can further improve inference accuracy [11], [12] and speed up loss-function value convergence for distributed learning [8]. Third, in addition

to migrating the data collected from UDs, the training-parameter exchange among edge nodes affects the learning speeds of AI models. For example, frequent model aggregation in federated learning leads to fast convergence at the cost of high data traffic volume among edge nodes.

Service Data Management

Given the potential impacts of data flow on the performance of AI services, service data management is required. For AI services, the following management conditions are necessary:

- *AI function placement*: The functions of an AI service can be executed at edge nodes. An AI function-placement policy determines which edge nodes are selected to host AI functions. An example of AI function placement is illustrated in Figure 1. In the inference phase, APs 1–3 and the macro BS (MBS) provide inference for UDs for AI service 2. In the model-training phase, APs upload and download AI models to/from small BS 2 to train the AI model in AI function 4. By placing AI functions at edge nodes, data flow among edge nodes can be initialized, and the learning structure for AI services can be defined.
- *The parameter selection of AI models*: The parameters in an AI model can be learning rate for DNNs and model-aggregation frequency in federated learning. Based on AI function placement, edge nodes train the AI models adopted by AI functions based on the parameters of the AI models, and thus, the parameters affect AI service performance, e.g., accuracy and learning speed. Moreover, they specify the frequency of parameter exchange and the amount of data for parameter exchange among edge nodes over time.
- *AI service operation*: The AI service operation is responsible for scheduling data flow in real time according to network conditions, such as channel conditions and instantaneous computing latency of edge nodes, given AI function placement and AI model parameters. For inference, the AI service-operation policy generates a realtime routing strategy for fast UD data uploading and processing among edge nodes. For model training, the AI service-operation policy determines whether, where, and how to migrate data among edge nodes for achieving data load balancing and improving AI service performance.

Network Slicing for AI Services

Connection With Resource Management

Network resources should be properly allocated to support service data management for both inference and model training. Service data management consumes communication and computing resources for exchanging data among edge nodes and processing data on edge nodes. High communication latency in data transmission or high computation latency in processing degrades AI service performance. Therefore, proper service data management should balance communication and computing resource consumption at each edge node to avoid bottlenecks in data delivery, processing, and training. It is necessary to jointly manage data and resources to support AI service provisioning.

Overview

In network slicing, a software-defined networking controller is deployed in the network to create and manage slices for different services and allocates virtual network resources accordingly. Specifically, network resources are first reserved for slices—referred to as *resource reservation*—based on service requirements, and subsequently allocated to individual UDs in real time, referred to as *resource scheduling* [13]. Although network slicing can support general computing services, further innovations are necessary to support AI services due to their unique features and requirements, as discussed in the “AI Services and Requirements” section. In the following subsections, we first discuss the challenges that network slicing faces in supporting AI services. Then,

to cope with the challenges, we propose a novel resource-pooling method, which is customized for AI services, to refine resource virtualization. Finally, we present an approach for AI service provisioning by integrating service data management into network slicing.

Challenges in AI Service Provisioning

As mentioned in the "Connection With Resource Management" section, data availability at edge nodes impacts the performance of an AI service, and improving data availability via data migration consumes network resources. Existing network slicing solutions allocate resources without taking service data management into account. Without the coordination of data flow, network slicing cannot satisfy the service requirements unique to AI services.

The location of physical resources affects the performance of AI services at the network edge. If computing units in an edge node far away from a UD are selected for inference or training, a long inference latency or a slow learning speed may occur due to multihop communications. Additionally, because of uneven UD spatial distribution, edge nodes at different locations may receive different amounts of data and learn at different speeds. Exchanging service data and learning models among edge nodes can improve AI service performance, and the location of edge nodes can impact the efficiency of data exchange and model training. In network slicing, taking the physical resource location into consideration complicates resource allocation and network operation, especially for network function placement and routing.

To address these two challenges, here we propose a resource-pooling method to refine the conventional resource virtualization method in network slicing. The objective is to customize resource virtualization for each AI service according to the location of physical resources and to allocate network resources while considering data migration among edge nodes.

Resource Pooling for AI Services

Physical resources in a network can be abstracted to a virtual resource pool via resource pooling, as shown in Figure 2. In the virtual resource pool, virtual APs (VAPs) represent logical servers with computing and storage capabilities and are connected by logical links. Edge nodes, equipped with computing units and storage, are projected to VAPs in the pool. Virtual network functions (VNFs), as the software implementation of service functions including both AI and conventional functions, are placed at the VAPs. A VAP can accommodate multiple VNFs, supported with proper virtual resources for communication, computing, and information storage. The resource pool is referred to as a *primary resource pool*.

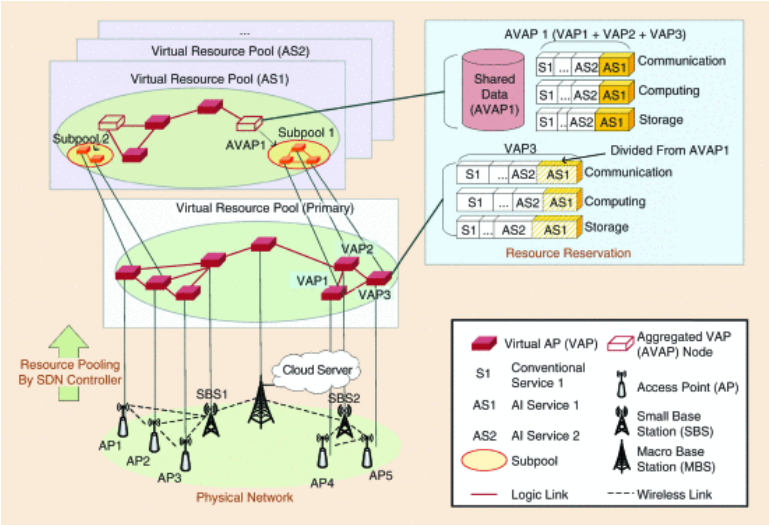

Figure 2 Resource pooling and reservation in network slicing.

Based on the primary resource pool and the physical location of edge nodes, we further abstract physical network resources into customized virtual resource pools, referred to as *secondary resource pools for individual AI services*. VAPs that support one VNF can form a subpool to facilitate resource and data sharing for that VNF. Correspondingly, VAPs are aggregated as an aggregated VAP (AVAP) for that VNF in the secondary resource pool. An example of a secondary resource pool is illustrated in Figure 2, where subpools are formed by VAPs 1–3 for a VNF of AI service 1 (AS1). Within a subpool, the data collected by VAPs can be migrated among VAPs for inference or model training.

An AVAP consists of all resources of the VAPs in the corresponding subpool. During resource reservation, the resources in both the AVAP and VAPs are reserved. Specifically, the resources of an AVAP are first reserved for a VNF to satisfy service requirements. The reservation should account for necessary resources for inference, model training, and data migration among the VAPs within a subpool. Then, the VAPs in the subpool of the AVAP can flexibly share the resources allocated at the AVAP. In the example shown in Figure 2, VAPs 1–3 are aggregated as AVAP1 in AS1. These VAPs reserve resources for AS1 as long as their reserved resources do not exceed the overall resources reserved for AS1 allocated at AVAP1. Although secondary resource pools are used for AI services, conventional services can reserve resources from the primary virtual resource pool. In the aforementioned example, the resources at VAP3 are reserved for all services, including both conventional and AI ones. During resource scheduling, when reserved resources at a VAP are not sufficient to support inference or training, data from UDs can be migrated to other VAPs within the same subpool for inference or training.

The main idea of the proposed resource-pooling method is to aggregate the resources of VAPs to adjust the learning structure of edge intelligence and balance the amount of data available to VAPs. The goal is to enable service data management in network slicing for satisfying AI service requirements. The resource-pooling policy depends on AI function placement, the geographical distribution of UDs in the network, and location of physical resources. First, AI function placement determines which VAPs have the same VNF and thus can be aggregated. Then, the geographical distribution of UDs and location of physical resources determine the amount of data that can be collected by each VAP. Accordingly, the data available to VAPs can be balanced by migrating data among edge nodes in a subpool. Finally, the geographical distribution of UDs and location of physical resources further affect the amount of network resources consumed in VAP aggregation. Specifically, VAP aggregation requires additional communication resources to enable data migration among VAPs in a subpool and computing resources for training the data within the subpool [14].

Service Provisioning for AI Services

Our AI service-provisioning approach combines the service data management techniques discussed in the “Service Data Management” section and the resource-pooling method mentioned in the “Resource Pooling for AI Services” section. We illustrate the approach in Figure 3.

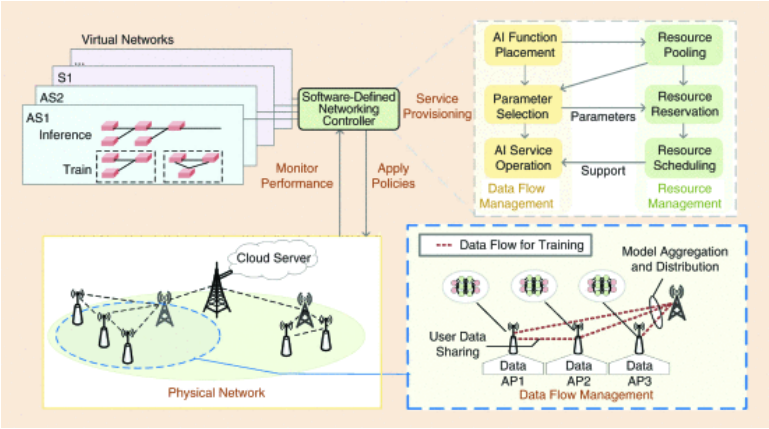

Figure 3 Service provisioning for AI services, which integrates network slicing and AI service management.

A software-defined networking controller is deployed in the network to manage network resources for all network services, including AI ones. First, AI functions and their corresponding VNFs are placed on edge nodes and their corresponding VAPs, respectively. AI function-placement policies are adjusted on a large time scale, e.g., days or hours. Furthermore, according to the physical location of network resources, the geographical distribution of UDs, and AI function placement, secondary virtual resource pools are determined for AI services. AI model parameters are selected according to AI function placement and potential data migration within subpools, and resources in the VAP and AVAPs are reserved for different VNFs to meet service requirements. Note that resources for both inference and model training are reserved for the VNFs of AI services. The resource-pooling policy for AI services, the parameter selection of AI models, and resource reservation are adjusted on a medium time scale, e.g., hours or minutes, to accommodate the spatial-temporal variations of the geographical distribution of UDs. Finally, in real-time network operations, reserved resources are allocated to individual UDs and network edges according to UD and network dynamics, such as UD mobility and channel conditions. The data from UDs may migrate among VAPs within a subpool according to the real-time AI service operation policy—with support from network resource scheduling—to maximize resource utilization and satisfy service requirements. The policies of both resource scheduling and AI service operation are adjusted on a small time scale, e.g., seconds or milliseconds.

In the example shown in Figure 3, AI functions are deployed at AP1–AP3, where federated learning is adopted for training AI models in the functions. The VAPs, corresponding to AP1 and AP2, are in a subpool for sharing the data collected from UDs. Parameters, e.g., the frequency for model aggregation, are determined by the software-defined networking controller, and network resources on the VAPs are reserved and scheduled correspondingly. Note that AP1 and AP2 may train their local models together with data migration for eliminating data bias and improving AI service performance.

Case Study: Service-Oriented Resource Pooling

In the following section, we first present a learning-based method for determining a resource-pooling policy. Then we provide an experiment to demonstrate the effectiveness of resource-pooling policies.

Learning-Based Resource Pooling

As mentioned in the “Resource Pooling for AI Services” section, network resources are reserved and scheduled for AI services from secondary resource pools. With different resource-pooling policies, the structure of secondary resource pools and the resulting AI service performance are different. Therefore, as depicted in Figure 4, we utilize a learning module supported by machine learning techniques, e.g., DNNs, to learn the AI service performance and resource consumption corresponding to resource-pooling policies, given resource

allocation and service data management strategies. The inputs of the learning module include the AI function-placement policy, resource-pooling policy for all AI services, and geographical distribution of UDs during a time interval between two successive resource-pooling policy updates. The outputs are the performance and the average resource consumption of AI services during the time interval. The learning module is trained at the software-defined networking controller. Specifically, the software-defined networking controller deploys different resource-pooling policies, monitors corresponding AI service performance and resource consumption, and uses the monitored information to further train the learning module. When the learning module is fully trained, the software-defined networking controller selects resource-pooling policy candidates that yield satisfactory AI service requirements with minimum resource consumption. Then, an AI service provider chooses a resource-pooling policy from the candidates based on the service-specific criteria, and the software-defined networking controller deploys the selected policy in the network.

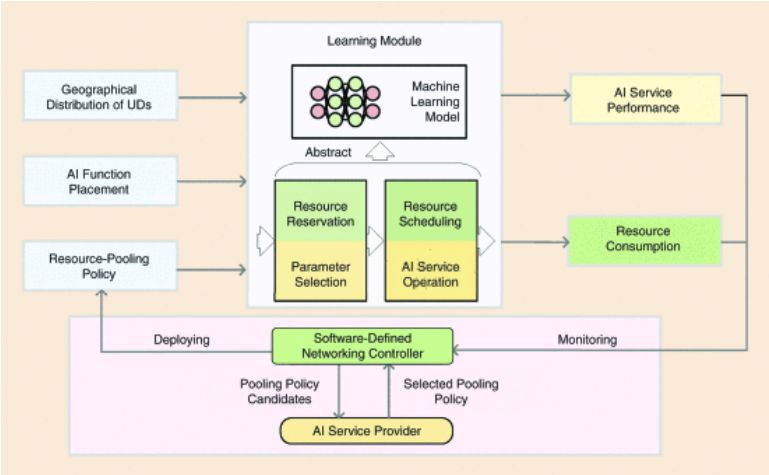

Figure 4 Service-oriented resource pool division.

Numerical Results

Experiment Setup

We conduct trace-driven simulations to evaluate the proposed resource-pooling method and determine the corresponding learning-based resource-pooling policy. In the considered network, there are 32 APs on the network edge. Each AP has deployed the same AI function for inference. AI models in the function are trained using a federated learning algorithm. Specifically, APs gather data from UDs and train their local models once every second. An MBS gathers the parameters of local models from APs once every 10 s, generates a global model using the FedAvg algorithm [8], and distributes the parameters of the global model to all the APs. The content of the AI function used in the simulation is handwritten digit recognition using a data set from the Modified National Institute of Standards and Technology database [15]. The AI model in a function includes three fully connected layers, with 784, 200, and 10 neurons, respectively. The learning rate for training the local model is 0.01, and the optimizer is stochastic gradient descent. In our simulation, the data collected by different APs is nonindependent identically distributed. We use a regression technique to implement the learning module, as mentioned in the “Learning-Based Resource Pooling” section. Specifically, we use a two-term Gaussian model with 95% confidence bounds and six different coefficients to regress the relationship between the number of subpools and AI service performance, i.e., the convergence rate and accuracy. The loss function for determining the Gaussian model is root-mean-square error.

We use different aggregated arrival rates of data for UDs at different APs. The data-arrival rate at an AP is randomly selected from $(0,\lambda max]$, where λmax denotes the maximum data-arrival rate. Each AP corresponds to

a VAP in the primary resource pool. We change the number of subpools in the secondary resource pool to adjust the pooling policy. The VAPs are grouped to form subpools according to the physical locations of the APs and the data-arrival rates at the APs by the *k*-means method. The resource requirements are summarized as follows: one resource unit (RU) is consumed for transmitting one unit of data between any two APs, 0.5 RU is consumed for processing one unit data for training, and 0.1 RU is consumed for offloading and distributing DNN models in federated learning. Moreover, 10 RUs are consumed for training a DNN model. The software-defined networking controller reserves resources accordingly based on an average data-arrival rate and schedules the resources. During resource scheduling, the additional cost is applied if reserved resources become insufficient.

Performance Evaluation

The impact of resource-pooling policy on AI service performance is displayed in Figure 5(a). By aggregating data into fewer APs, model training is conducted in a more centralized learning structure, and data bias can be eliminated by balancing the collected data among APs. As shown in Figure 5(a), compared to centralized learning with one subpool in the virtual resource pool, the accuracy and the convergence rate of loss function are reduced by 5 and 0.2%, respectively, when fully distributed learning with 32 subpools is adopted. A higher training speed and a higher level of accuracy can be achieved under a pooling policy with a lower number of subpools. Moreover, we utilize a learning module to model the relationship between the resource-pooling policy and AI service performance, as presented in the “Learning-Based Resource Pooling” section. As presented in Figure 5(a), the AI function performance approximated by the learning module is accurate. The resource consumption for training with different resource-pooling policies is shown in Figure 5(b). As the number of subpools decreases, model training requires more communication resources but less computing resources for training. This is because more APs migrate their collected data, which generates additional cost on communication, while fewer APs train their local models, which reduces the overall computing resource consumption.

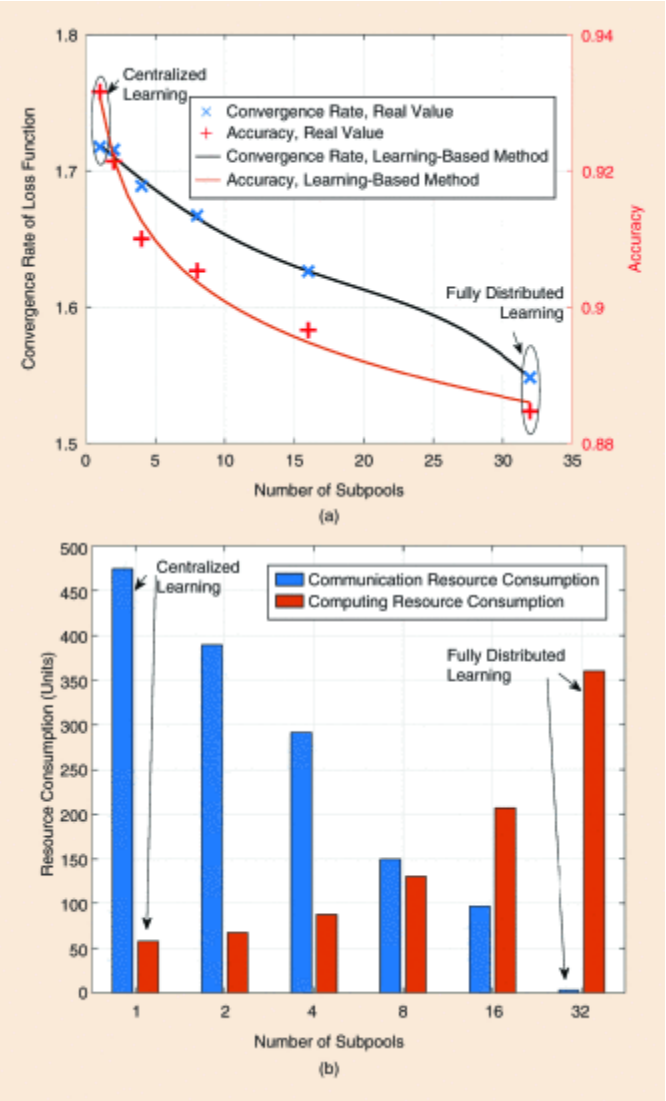

Figure 5 Service performance and resource utilization versus the number of subpools when λmax=1, where the accuracy in (a) is defined as the *fraction of correct inferences over all inferences*. (a) The accuracy and the convergence rate of loss of the AI function. (b) Resource utilization for model training in the AI service.

The AI service performance and average resource consumption with different user data-arrival rates, λmax, and resource-pooling policies are depicted in Figure 6. As λmax increases, the resource consumption increases due to the need for processing more data in training. Meanwhile, with a lower arrival rate, the accuracy of the AI service degrades. This is because the available data for training at each AP decreases, and overfitting happens when a small amount of data is trained with a high learning rate.

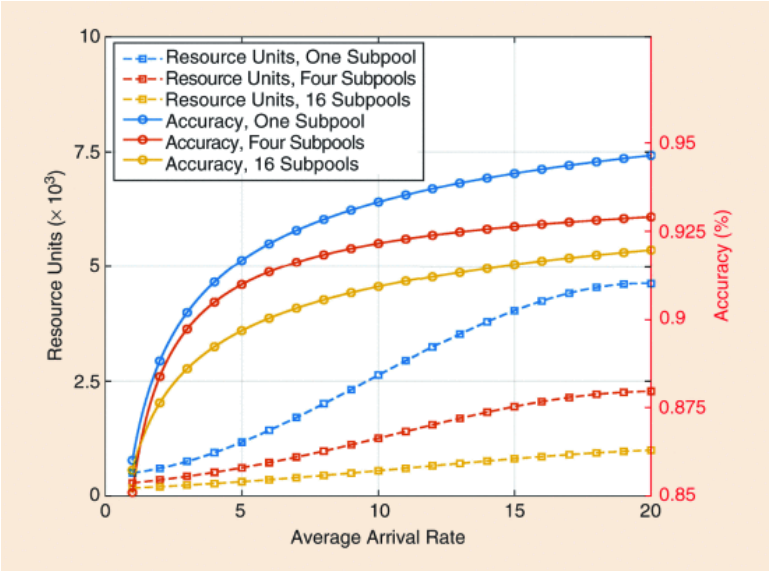

Figure 6 Service performance and average resource consumption versus user data-arrival rates.

Conclusions

In this article, we investigated AI service provisioning on the network edge for 6G. Because AI services depend on data for training and inference, AI service provisioning requires the joint management of data and conventional network resources. Accordingly, within the framework of network slicing, we proposed a resource-pooling method to connect data and network resources in AI service provisioning. The proposed method supports data management in network slicing while balancing AI service performance and resource consumption of data management. In addition, the proposed method considers the location of physical resources in resource virtualization for network slicing. With our approach, network and service providers can jointly determine where and how to train AI models based on data availability, network resource constraints, and service performance requirements.

References

1. X. Shen et al., "AI-assisted network-slicing based next-generation wireless networks", *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45-66, 2020.

2. W. Saad, M. Bennis and M. Chen, "A vision of 6G wireless systems: Applications trends technologies and open research problems", *IEEE Netw.*, vol. 34, no. 3, pp. 134-142, 2020.

3. W. Wu et al., "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning", *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2076-2089, 2021.

4. M. Li, J. Gao, L. Zhao and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks", *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122-1135, 2020.

5. J. Gao, M. Li, W. Zhuang, X. Shen and X. Li, "MAC for machine type communications in industrial IoT – Part II: Scheduling and numerical results", *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9958-9969, 2021.

6. H. H. Yang, Z. Liu, T. Q. S. Quek and H. V. Poor, "Scheduling policies for federated learning in wireless networks", *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 317-333, 2020.

7. J. Ren, G. Yu and G. Ding, "Accelerating DNN training in wireless federated edge learning systems", *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 219-232, 2021.

8. X. Li, K. Huang, W. Yang, S. Wang and Z. Zhang, "On the convergence of FedAvg on non-IID data", 2019.

9. Q. Cao, N. Balasubramanian and A. Balasubramanian, "MobiRNN: Efficient recurrent neural network execution on mobile GPU", *Proc. 1st Int. Workshop Deep Learn. Mobile Syst. Appl.*, pp. 1-6, 2017.

10. V. Vanhoucke, A. Senior and M. Z. Mao, "Improving the speed of neural networks on CPUs", *Proc. 24th Annu. Conf. Neural Inf. Process. Syst.*, pp. 1-8, 2011.
11. H. Wang, Z. Kaplan, D. Niu and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning", *Proc. IEEE Conf. Comput. Commun.*, pp. 1698-1707, 2020.
12. E. Ntoutsi et al., "Bias in data-driven artificial intelligence systems—An introductory survey", *Wiley Interdisciplinary Rev. Data Mining Knowl. Discovery*, vol. 10, no. 3, pp. e1356, 2020.
13. W. Zhuang, Q. Ye, F. Lyu, N. Cheng and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication computing and caching", *Proc. IEEE*, vol. 108, no. 2, pp. 274-291, Feb. 2020.
14. J. Liu, J. Liu, W. Du and D. Li, "Performance analysis and characterization of training deep learning models on mobile device", *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst.*, pp. 506-515, 2019.
15. Y. LeCun and C. Cortes, *The MNIST database handwritten digit database*, 2010, [online] Available: http://yann.lecun.com/exdb/mnist/.