Fine-Grained AI Model Caching and Downloading With Coordinated Multipoint Broadcasting in Multi-Cell Edge Networks

Yang Fu, Peng Qin, *Member, IEEE*, Yueyue Zhang, and Yifei Wang

***Abstract*—6G networks are envisioned to support on-demand AI model downloading to accommodate diverse inference requirements of end users. By proactively caching models at edge nodes, users can retrieve the requested models with low latency for on-device AI inference. However, the substantial size of contemporary AI models poses significant challenges for edge caching under limited storage capacity, as well as for the concurrent delivery of heterogeneous models over wireless channels. To address these challenges, we propose a fine-grained AI model caching and downloading system that exploits parameter reusability, stemming from the common practice of fine-tuning task-specific models from a shared pre-trained model with frozen parameters. This system selectively caches model parameter blocks (PBs) at edge nodes, eliminating redundant storage of reusable parameters across different cached models. Additionally, it incorporates coordinated multipoint (CoMP) broadcasting to simultaneously deliver reusable PBs to multiple users, thereby enhancing downlink spectrum utilization. Under this arrangement, we formulate a model downloading delay minimization problem to jointly optimize PB caching, migration (among edge nodes), and broadcasting beamforming. To tackle this intractable problem, we develop a distributed multi-agent learning framework that enables edge nodes to explicitly learn mutual influence among their actions, thereby facilitating cooperation. Furthermore, a data augmentation approach is proposed to adaptively generate synthetic training samples through a predictive model, boosting sample efficiency and accelerating policy learning. Both theoretical analysis and simulation experiments validate the superior convergence performance of the proposed learning framework. Moreover, experimental results demonstrate that our scheme significantly reduces model downloading delay compared to benchmark methods.**

***Index Terms*—Edge AI, model downloading, edge caching, CoMP broadcasting, multi-agent learning.**

I. Introduction

A. Background

THE rapid proliferation of edge computing resources, coupled with transformative breakthroughs in artificial intelligence (AI), has given rise to edge AI [1]. As a pivotal enabler of 6G networks, edge AI supports ubiquitous inference services through the distributed deployment of AI models across edge infrastructure and user devices [2], [3]. For inference tasks characterized by high data sensitivity and stringent privacy requirements, such as mobile health and virtual assistants, on-device AI inference is often mandatory to preserve all data at the user side [4]. However, the substantial storage demands of contemporary AI models (e.g., Apple's on-device large language model (LLM) OpenELM, which comprises 3 billion parameters and requires 12 GB of storage) render it impractical for user devices with limited capacity to store all necessary models locally [5]. A viable solution is to adaptively download AI models from the network[1], accommodating real-time and diverse inference needs while preventing excessive local storage consumption.

Indeed, on-demand model downloading from an AI repository (which comprises a large collection of trained models and is typically deployed in cloud centers) has been identified as a key use case in the standardization of 6G [6], [7]. According to 3GPP technical specifications, such downloading operations should be completed within seconds for general inference tasks, and within 10-100 ms for time-sensitive applications including humanoid robot control and autonomous driving [7], [8]. To satisfy these stringent delay requirements, caching AI models at edge nodes near users is essential for eliminating the unpredictable delays inherent in cloud access [9]. Meanwhile, advanced wireless transmission techniques (particularly broadcasting and coordinated multipoint (CoMP) considered in this paper) should be exploited to boost downlink throughput, thereby accelerating model delivery from edge nodes to end users. Building upon these two directions, research efforts have been dedicated to facilitating AI model caching and downloading in edge networks.

B. Related Works

1) Works on Edge AI Model Caching: Given the limited storage capacity of edge nodes, only a finite number of AI models can be cached from the AI repository. Therefore, existing works primarily focus on developing optimal model selection strategies at edge nodes to align with user requests. Literature [10] investigated a single edge node scenario, where a subset of AI models was cached by the edge node to execute inference tasks. The authors proposed a deep reinforcement

This work was supported in part by 62201212, 62271201, 2023YFB2904700, and MPIS202409. *(Corresponding author: Peng Qin)*

Yang Fu, Peng Qin, Yifei Wang are with the State Key Laboratory of Alternate Electrical Power System with Renewable Energy Sources, School of Electrical and Electronic Engineering, North China Electric Power University, Beijing 102206, China (e-mail: qinpeng@ncepu.edu.cn).

Yueyue Zhang is with the China Satellite Network Group Shanghai Research Institute, Shanghai 200131, China.

[1]Accomplishing AI inference through task offloading to edge/cloud servers and downloading models to local devices represent two complementary rather than substitutable paradigms. This work focuses on the latter, motivated by its advantages in stable low-latency inference without repeated network dependency, enhanced data security, improved location and context awareness, and stronger sustainability for continuous tasks.

learning (DRL)-based approach to train the caching policy while introducing penalty mechanism to avoid frequent model change. Paper [11] jointly optimized the number and type of AI models to be cached at an edge node, so as to provide scalable inference service. A rounding and relocation method was developed to handle the integer constraints of caching variables, thereby minimizing the holistic cost and accuracy loss. The authors of [12], [13] elaborated the cooperative caching in multi-cell scenarios, in which AI models could be migrated among edge nodes to enhance the cache hit ratio. Multi-agent DRL (MADRL) was adopted to make distributed caching decisions, while promoting model routing via backhaul links. In [14], different compressed versions of an AI model were deployed at cloud-edge-device nodes according to their computational capacities. The authors developed a DRL algorithm integrated with numerical optimization to derive the task offloading and resource allocation solution, thereby balancing inference delay, energy consumption, and accuracy loss. However, the aforementioned studies follow conventional principles of content caching, which directly stores entire AI models at edge nodes. Given that AI models typically require significantly greater storage space than traditional cached content, such coarse-grained caching methods are inefficient in utilizing the limited edge storage capacity.

Some works proposed storage-efficient caching strategies by exploiting the architectural properties of AI models, particularly deep neural networks. Layer splitting was considered in [15] to determine the number of model layers to be cached at each edge node, then nodes storing different layers sequentially execute inference tasks through intermediate result exchange. Paper [16] employed semantic splitting to partition the AI model into parallel disjoint fragments. Online learning was adopted to cache model fragments at different edge nodes, thereby reducing energy consumption and response time. Nevertheless, the methods proposed in [15], [16] are limited to caching a single AI model with a specific architecture. They cannot scale to the concurrent deployment of a large number of heterogeneous models from the AI repository, failing to meet diverse model downloading demands. To fill this gap, our work exploits not only the internal structure of individual AI models to partition them into fine-grained parameter sets, but also the interplay among different models by identifying their overlapping parameters for scalable edge caching.

2) Works on Broadcasting and CoMP: By leveraging the broadcast nature of wireless channels, edge node can simultaneously deliver identical data to multiple users, thereby improving spectral efficiency. To mitigate inter-cell interference, multiple nodes can further form a CoMP cluster for joint data transmission, which effectively enhances the downlink throughput [17]. Reference [18] investigated the hybrid beamforming in a CoMP broadcasting scenario, in which the optimal precoder was derived by semi-definite program, then a maximum ratio combiner was employed to maximize the received signal power. In [19], a two-stage optimization approach was designed to maximize the quality of service (QoS) of data broadcasting, in which cells were first clustered to reduce inter-cell interference, then the service order was scheduled to decrease the total data reception delay. The authors of [20] considered an air-ground communication system where multiple drones broadcast data to user groups using CoMP. To perform system optimization, QMIX was invoked to output the drone flight direction, then the CoMP beamforming was optimized using majorization minimization algorithm. Literature [21] elaborated a multi-cell integrated sensing and communication system with CoMP, where the inter-cell reflections could be utilized to enhance the target estimation performance owing to the sharing of edge nodes' data. The authors proposed robust CoMP beamforming approach taking into account imperfect channel state information (CSI). In the context of edge AI, broadcasting has been employed to distribute inference tokens across devices, enabling the parallel execution of mixture-of-experts foundation models [22]. In [23], user-offloaded tasks are jointly processed by multiple edge nodes that form a CoMP cluster for data exchange and result delivery, where the success probability is maximized to enhance both communication and computation efficiency. Although existing studies have demonstrated the effectiveness of broadcasting and CoMP techniques for enhancing data rates, their seamless integration with AI model downloading remains an open issue. Unlike traditional multimedia content, AI models are often trained for various downstream tasks, leading to low information overlap among user requests. Furthermore, the above works assume that all user data is readily available at edge nodes to enable CoMP, while overlooking the data exchange overhead. For AI models with large data volumes, excessive migration among edge nodes may incur severe downloading delays and backhaul congestion.

C. Motivations and Contributions

To address the limitations of previous studies, we propose a novel fine-grained AI model caching and downloading (FGAMCD) system that exploits parameter reusability among AI models. This property arises from the common practice of fine-tuning task-specific models from a shared pre-trained model, inherently maintaining a set of reusable parameters across different AI models [24], [25]. For instance, in convolutional neural networks (CNNs), the shallow layers responsible for extracting common visual features typically exhibit high parameter reusability across different tasks. This effect is even more pronounced in emerging LLMs, where parameter-efficient fine-tuning (PEFT) techniques (e.g., low-rank adaptation) typically freeze over 99% of pre-trained parameters, resulting in a significant proportion of reused parameters [26].

In the proposed FGAMCD system, edge nodes selectively cache parameter blocks (PBs), which are fine-grained components of AI models, such as neural network layers or Transformer blocks. When multiple AI models are cached, the edge node only needs to store a single copy of their reusable PBs, thereby enhancing storage utilization. Additionally, it integrates broadcasting to simultaneously deliver PBs to multiple users, leveraging potential PB reuse across different requested models. To further reduce the downloading delay, FGAMCD optimizes PB migration within the backhaul network, facilitating the CoMP transmission of edge nodes while balancing coordinated gains and data exchange overhead.

Parameter reusability has also been explored in prior works to support AI model caching and delivery. The TrimCaching framework proposed in [27] optimizes edge caching of AI models by considering parameter sharing among models, thereby improving cache hit ratio. However, this approach orthogonally allocates downlink spectrum to each user and relies on traditional unicasting for model delivery, leaving room for downloading delay reduction. Moreover, the proposed heuristic caching method incurs high computational complexity, limiting its adaptability to dynamic user requests and communication conditions. Work in [28] developed a model broadcasting protocol to serve multiple users performing heterogeneous inference tasks, along with the joint optimization of parameter selection and power control. Although this protocol incorporates broadcasting of reused PBs, the proposed approach cannot be directly extended to multi-cell and multi-antenna communication systems, which require dedicated PB migration and beamforming design. To the best of our knowledge, this is the first work to jointly optimize AI model caching, migration, and delivery in multi-cell edge networks, through a backhaul-wireless co-design for communication-efficient model downloading. In particular, the combination of caching-aware PB migration with CoMP broadcasting unleashes the full potential of parameter reusability, offering simultaneous improvements in storage and spectrum efficiency, which remain only partially addressed in [27], [28]. The main contributions are summarized below.

1) We propose an FGAMCD system to fulfill diverse downloading requests by strategically caching PBs from an AI model repository across multiple edge nodes, enabling efficient model retrieval for end users. Subsequently, the edge nodes cooperatively deliver the cached PBs to users through CoMP broadcasting, while PBs are migrated within backhaul links to facilitate coordinated transmission and enhance downlink rates. After receiving all the required PBs, each user reconstructs the complete model to perform on-device AI inference. With the objective to minimize the total model downloading delay, we formulate a joint optimization problem of PB caching, migration, and broadcasting beamforming, subject to user QoS requirements, edge storage capacity and transmission power constraints.
2) The formulated problem is an intractable mixed-integer nonlinear programming (MINLP) problem, exacerbated by the substantial computational complexity of centralized control across multiple edge nodes. To solve this problem, we develop a distributed MADRL framework, named multi-agent action semantics network with data augmentation (MAASN-DA), which learns PB caching and migration policies by explicitly characterizing the influence of each edge node's action on others through a specialized actor network. In addition, MAASN-DA adaptively generates synthetic training samples through a predictive model to enrich the experience replay buffer, thereby accelerating policy learning. A robust optimization subroutine for deriving CoMP broadcasting beamformers under imperfect CSI is proposed and incorporated into the reward calculation, thus completing the MAASN-DA training loop.
3) We provide a theoretical convergence analysis for MAASN-DA, deriving a closed-form upper bound on the Q-value approximation error to offer theoretical guidelines for learning hyperparameter configuration. Furthermore, extensive experiments utilizing real-world dataset and AI models are conducted. The ablation study demonstrates that the proposed MAASN-DA achieves higher cumulative reward and faster convergence speed than existing MADRL methods. Moreover, experimental results reveal a two-fold performance gain achieved by the FGAMCD system: *i) caching efficiency gain* by eliminating redundant storage of reusable PBs, and *ii) downloading efficiency gain* through coordinated PB broadcasting that simultaneously serves multiple user requests. Compared with conventional coarse-grained caching and unicasting-based model delivery, FGAMCD reduces model downloading delay by 29.74% to 67.86%.

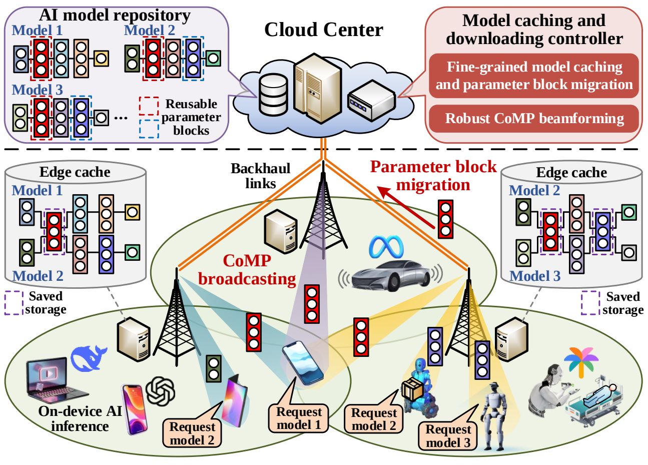


Fig. 1. Fine-grained AI model caching and downloading system model.

Notations: Superscripts H and T denote the conjugate transpose and transpose, respectively. $\|\cdot\|$ indicates the norm of a vector or spectral norm of a matrix. $\mathbb{C}^{M\times N}$ specifies the space of $M\times N$ complex matrices, and $\mathbb{R}$ is the set of real numbers. $\{x\}^+$ is equal to $\max\{x,0\}$. $\mathbf{X}\succeq 0$ means that $\mathbf{X}$ is a semi-definite matrix. $\mathbf{O}$ denotes the all-zero matrix.

II. System Model and Problem Formulation

We consider a multi-cell network architecture as illustrated in Fig. 1, designed to cache popular AI models[2] across N distributed edge nodes, which then provide model downloading service to U end users. The sets of edge nodes and users are signified by $\mathcal{N}=\{1,\ldots,n,\ldots,N\}$ and $\mathcal{U}=\{1,\ldots,u,\ldots,U\}$, respectively. The cloud center maintains a comprehensive AI model repository $\mathcal{J}=\{1,\ldots,j,\ldots,J\}$ comprising J models, which can be downloaded by users to

[2]To mitigate the risk of sensitive information leakage or adversarial attacks arising from model distribution across edge nodes, the proposed FGAMCD framework can be integrated with ownership protection mechanisms such as model watermarking [29]. In this case, each edge-cached model is embedded with a watermark (e.g., a binary vector derived from model parameters), enabling users to verify the authenticity of the downloaded model.

TABLE I
SUMMARY OF MAIN NOTATIONS

Notation	Description
N, $\mathcal{N}$	Number and set of edge nodes
U, $\mathcal{U}$	Number and set of users
J, $\mathcal{J}$	Number and set of AI models
K, $\mathcal{K}$	Number and set of PBs
r_u	Target AI model of user u
$\mathcal{K}_j$	Set of PBs that constitute AI model j
$a_n(k)$	Binary variable indicating whether node n caches PB k
$S(k)$	Data size of PB k
C_n	Storage capacity of node n
$b_{n,m}(k)$	Binary variable indicating whether to migrate PB k from edge node n to m
$\lambda_n(k)$	Binary indicator representing whether edge node n participates in the delivery of PB k
M	Number of antennas at each edge node
$\mathbf{h}_{n,u}(k)$	Actual channel between node n and user u for PB k
$\tilde{\mathbf{h}}_{n,u}(k)$	Estimated channel between node n and user u for PB k
$\mathbf{w}_n(k)$	Beamforming at edge node n for broadcasting PB k
$R_u(k)$, Q_u	Downloading rate and QoS requirement for user u
$R^{\text{bac}}_{n,m}(k)$	Backhaul link rate between edge node n and m
$T(k)$, T	Downloading delay for PB k and total downloading delay
$\mathbf{o}_n(k)$, $\mathbf{s}(k)$	Observation of agent n and global state in step k
$\mathbf{d}_n(k)$, $r(k)$	Action of agent n and reward in step k
$\boldsymbol{\eta}_{\text{in}}, \boldsymbol{\eta}_{\text{re}}, \boldsymbol{\eta}_{\text{out}}$	Weighting parameters of ESN
$\mathbf{q}(k)$	Reservoir state of ESN in step k
ξ, τ_0	Selection threshold and proportion for synthetic samples
$\boldsymbol{\varphi}_n$	Parameters of agent n's action semantics actor network
$\boldsymbol{\theta}_n, \boldsymbol{\theta}^{\text{mix}}$	Parameters of agent n's critic and mixing network
γ, E	Discount factor and number of training episodes

facilitate on-device AI inference. All edge nodes are interconnected with the cloud server via backhaul links. Main notations to be used are summarized in Table I. The system operation encompasses the following phases:

- **User Model Request:** Each user u submits a model request to its associated edge nodes, specifying both the target AI model $r_u \in \mathcal{J}$ from the repository and the corresponding QoS requirement Q_u for model downloading.
- **Fine-Grained AI Model Caching:** Edge nodes strategically cache a selected set of AI models from the cloud center, enabling collaborative fulfillment of user requests. Particularly, the proposed system exploits the parameter shareability, where distinct AI models may contain reusable PBs due to the prevalent adoption of parameter-efficient fine-tuning techniques. Accordingly, a fine-grained model caching mechanism is implemented to cache specific PBs instead of entire AI models at each edge node. This design ensures that the reusable PB is stored only once in the edge cache, thereby significantly enhancing storage efficiency. More details will be elaborated in Section II-A.
- **Model Downloading With CoMP Broadcasting:** During this phase, users retrieve the requested AI models from edge nodes through a coordinated transmission process. Specifically, edge nodes utilize backhaul links to exchange their cached PBs, promoting the execution of CoMP techniques. Subsequently, they cooperatively broadcast PBs to the requesting users, exploiting potential PB reuse across different requested models to improve downloading efficiency. We will detail the model downloading phase in Section II-B to D.
- **On-Device AI inference:** End users reconstruct AI models from the received PBs to perform various inference tasks, e.g., text generation for virtual assistance applications and object recognition for humanoid robot control[3]. This local inference fashion inherently guarantees data privacy and security by eliminating the need for sensitive data/feature uploading.

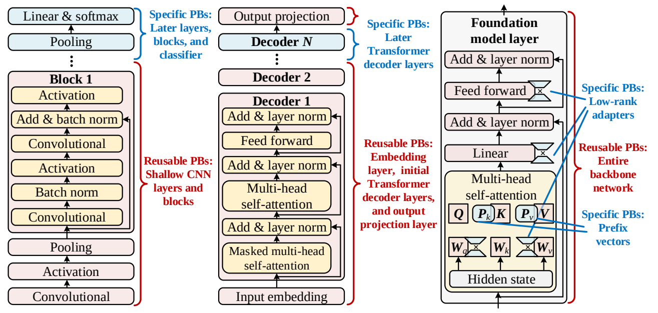

Linear & softmax
Pooling
Block 1
Activation
Add & batch norm
Convolutional
Activation
Batch norm
Convolutional
Pooling
Activation
Convolutional
Specific PBs: Later layers, blocks, and classifier
Reusable PBs: Shallow CNN layers and blocks
Output projection
Decoder N
Decoder 2
Decoder 1
Add & layer norm
Feed forward
Add & layer norm
Multi-head self-attention
Add & layer norm
Masked multi-head self-attention
Input embedding
Specific PBs: Later Transformer decoder layers
Reusable PBs: Embedding layer, initial Transformer decoder layers, and output projection layer
Foundation model layer
Add & layer norm
Feed forward
Add & layer norm
Linear
Multi-head self-attention
Q
P_k
K
P_v
V
W_q
W_k
W_v
Hidden state
Specific PBs: Low-rank adapters
Reusable PBs: Entire backbone network
Specific PBs: Prefix vectors
(a) CNN (b) Decoder-only LLM (c) Foundation model with PEFT

Fig. 2. Identification of PBs across AI model architectures.

A. Fine-Grained AI Model Caching

For each AI model j in the repository, we define $\mathcal{K}_j$ as the set of PBs that constitute the model. $\mathcal{K} = \bigcup_{j\in\mathcal{J}} \mathcal{K}_j = \{1, \ldots, k, \ldots, K\}$ is the collection of all PBs[4], and we have $|\mathcal{K}| \le \sum_{j\in\mathcal{J}} |\mathcal{K}_j|$ due to the parameter shareability. The fine-grained caching decision is represented by binary variable $a_n(k) \in \{0,1\}$, where $a_n(k) = 1$ if PB k is cached by edge node n, otherwise $a_n(k) = 0$. Therefore, $\prod_{k\in\mathcal{K}_j} a_n(k) = 1$ implies that model j is completely cached by node n. Consider that the total cached PBs cannot exceed the storage capacity of each edge node, we have constraint[5]

$$\sum_{k\in\mathcal{K}} a_n(k)\, S(k) \le C_n,\ \forall n, \tag{1}$$

where $S(k)$ is the size of PB k, C_n signifies the storage capacity of node n.

Remark 1 (Identification of PBs Across AI Model Architectures): The definition of a PB depends on the underlying model architecture and parameter reuse conditions. As illustrated in Fig. 2, in CNNs, a PB can correspond to a single layer (e.g., convolution, activation, or linear) or a block containing residual connections or inception modules. Reusable PBs are often shallow layers or blocks that capture generic visual features. For decoder-only LLMs, a PB may

[3]After downloading, model reconstruction simply loads the PBs into their designated positions within the model architecture, without requiring retraining or fine-tuning. The associated overhead is negligible compared to PB downloading [30]. Moreover, this process does not affect inference accuracy, as it restores the original model exactly without modifying its architecture or parameter values.

[4]The integration of FGAMCD with model compression is straightforward. Diverse sub-models can be derived from an original model through techniques such as quantization, pruning, and knowledge distillation [6]. The PBs of these sub-models can be also incorporated into $\mathcal{K}$ without affecting our subsequent modeling and solution, thereby allowing users to flexibly select appropriate model versions to improve downloading efficiency.

[5]In practice, each edge node maintains an index table of PB identifiers, sizes, and locations to support lookup according to user requests. Since such information requires only minimal storage and thus negligible compared to cached PBs themselves, the indexing overhead is omitted [10], [11].

refer to the input embedding, a Transformer decoder layer, or the output projection (which typically shares parameters with the embedding). In this case, reusable PBs are usually the embedding/output layers and the initial decoder layers that extract common linguistic knowledge. In foundation models with PEFT, the backbone network itself can serve as a reusable PB across different downstream tasks, while the inserted fine-tuning parameters (e.g., low-rank adapters or prefix vectors) are treated as task-specific PBs. For models without clear parameter sharing, we directly treat them as individual PBs.

B. Parameter Block Migration

Prior to model delivery over the wireless edge network, edge nodes can exchange the cached PBs via backhaul links. This exchange enables multiple nodes to cooperatively transmit PBs to users, thereby enhancing the CoMP performance at the expense of increased backhaul delay. To achieve an optimal tradeoff, we introduce PB migration variable $b_{n,m}(k) \in \{0,1\}$, where $b_{n,m}(k) = 1$ indicates that PB k cached by edge node n is migrated to node $m \in \mathcal{N} \backslash \{n\}$, otherwise $b_{n,m}(k) = 0$. Obviously, a PB can be migrated from edge node n to other nodes only when it is cached by node n, thus we impose the following restraint:

$$a_n(k) \geq \max_{m \in \mathcal{N} \backslash \{n\}} \{b_{n,m}(k)\}, \ \forall k, n. \tag{2}$$

As a result, edge node n can participate in the delivery of PB k if at least one condition is met: i) PB k is already in node n's cache; ii) PB k is migrated to node n from another node. Formally, we define binary indicator $\lambda_n(k) \in \{0,1\}$ with $\lambda_n(k) = 1$ implying node n's participation in the delivery of PB k, then $\lambda_n(k)$ is calculated by

$$\lambda_n(k) = \min\left\{a_n(k) + \sum_{m \in \mathcal{N} \backslash \{n\}} b_{m,n}(k), 1\right\}. \tag{3}$$

Afterwards, $\lambda_n(k)$ will be leveraged to formulate the received signal for users during PB broadcasting.

C. Parameter Block Broadcasting

Each edge node equips with M antennas to deliver PBs to single-antenna users. Denote $\mathbf{h}_{n,u}(k) = \sqrt{\upsilon d_{n,u}^{-\alpha}(k)}\bar{\mathbf{h}}_{n,u}(k)$ as the wireless channel from edge node n to user u when broadcasting PB k, υ is the channel gain at reference distance of 1 m, $d_{n,u}(k)$ represents the transmission distance, α indicates the pathloss exponent, and $\bar{\mathbf{h}}_{n,u}(k)$ is the small-scale fading component. In practice, edge nodes face challenges in acquiring perfect CSI. Hence, we model the relationship between actual channel $\mathbf{h}_{n,u}(k)$ and estimated channel $\tilde{\mathbf{h}}_{n,u}(k)$ considering CSI uncertainty as follows

$$\begin{aligned} \mathbf{h}_{n,u}(k) &= \tilde{\mathbf{h}}_{n,u}(k) + \mathbf{e}_{n,u}(k), \\ \mathbf{e}_{n,u}(k) &\in \mathcal{E}_{n,u} = \left\{\mathbf{e} | \mathbf{e}^{\mathrm{H}} \mathbf{C}_{n,u} \mathbf{e} \leq 1\right\}, \end{aligned} \tag{4}$$

where $\tilde{\mathbf{h}}_{n,u}(k) \in \mathbb{C}^{M \times 1}$ denotes the estimated channel vector, which can be acquired either through pilot signal reception from the user or generated by a diffusion model [31]. This novel approach in [31] exploits the user location as conditional input and infers the statistical CSI from random noise via a reverse denoising process. $\mathbf{e}_{n,u}(k)$ represents the channel estimation error, which is restricted within a spherical set with $\mathbf{C}_{n,u}$ determining the shape and size of the spherical set.

PBs are sequentially broadcasted from edge nodes to users through CoMP transmission. Consider the broadcasting of an arbitrary PB, say PB k, the transmitted symbol of edge node n is denoted by $\lambda_n(k) x(k)$ with $\mathbb{E}\{|x(k)|^2\} = 1$. Then, the signal received at user u can be written as

$$y_u(k) = \sum_{n \in \mathcal{N}} \mathbf{h}_{n,u}^{\mathrm{H}}(k) \mathbf{w}_n(k) \lambda_n(k) x(k) + z_u(k), \tag{5}$$

where $\mathbf{w}_n(k) \in \mathbb{C}^{M \times 1}$ signifies the transmission beamforming at edge node n, $z_u(k)$ is the channel noise, which obeys a complex Gaussian distribution with zero mean and variance σ_u^2. Consequently, the downloading rate for user u during the broadcasting of PB k is given by

$$R_u(k) = B \log_2\left(1 + \frac{\left|\sum_{n \in \mathcal{N}} \lambda_n(k) \mathbf{h}_{n,u}^{\mathrm{H}}(k) \mathbf{w}_n(k)\right|^2}{\sigma_u^2}\right), \tag{6}$$

where B is the network bandwidth shared by all edge nodes.

D. Model Downloading Delay

The model downloading delay measures the total time required for all users to successfully retrieve the PBs necessary for assembling their requested AI models. Given that all PBs are delivered sequentially, we first derive the downloading delay for a specific PB k (with data size $S(k)$) in the sequel[6]

$$\begin{aligned} T(k) = &\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{N} \backslash \{n\}} \frac{b_{n,m}(k) S(k)}{R_{n,m}^{\mathrm{bac}}(k)} \\ &+ \max_{u \in \mathcal{U}} \frac{\mathbb{I}\{k \in \mathcal{K}_{r_u}\} S(k)}{\min\limits_{\mathbf{e}_{n,u}(k) \in \mathcal{E}_{n,u}, \forall n} R_u(k)}, \end{aligned} \tag{7}$$

where the first term of the right-hand-side quantifies the migration delay, and $R_{n,m}^{\mathrm{bac}}(k)$ denotes the backhaul link rate[7] from edge node n to m when delivering PB k. The second term is the worst case broadcasting delay, where $\mathbb{I}\{x\} = 1$ if x is true, otherwise $\mathbb{I}\{x\} = 0$, thus $\mathbb{I}\{k \in \mathcal{K}_{r_u}\}$ indicates whether u is the requesting user for PB k. On this basis, we express the total model downloading delay as

$$T = \sum_{k \in \mathcal{K}} T(k). \tag{8}$$

E. Problem Formulation

Our objective is to minimize the model downloading delay while satisfying both edge storage capacity constraints

[6]Similar to [11], [27], we do not account for the model fetching delay from the cloud to edge nodes for two reasons. First, model placement at the edge typically occurs at a much longer timescale than user downloading. Once cached, user requests only trigger PB migration and wireless delivery, which dominate the downloading delay. Second, our work focuses on users served by collaborative edge nodes to explore the performance gain of fine-grained caching, while the traditional cloud-edge-device model delivery lies beyond the scope of this study.

[7]Note that $R_{n,m}^{\mathrm{bac}}(k)$ is finite and varies across different k due to dynamic backhaul bandwidth availability and congestion conditions, introducing additional delay for PB migration. Our subsequent goal is to jointly design caching and migration strategies to minimize the overall downloading latency.

and user QoS requirements, thereby boosting timely on-device AI inference. The designed variables incorporate fine-grained model caching $\mathbf{a} = [a_n(k) : \forall k, n]$, PB migration $\mathbf{b} = [b_{n,m}(k) : \forall k, n, m]$, and broadcasting beamforming $\mathbf{w} = [\mathbf{w}_n(k) : \forall k, n]$. As a consequence, the optimization problem is formulated as

$$\mathbf{P1}: \min_{\mathbf{a},\mathbf{b},\mathbf{w}} T, \tag{9a}$$

$$\text{s.t.} \min_{\mathbf{e}_{n,u}(k)\in\mathcal{E}_{n,u},\forall n} R_u(k) \geq \mathbb{I}\{k \in \mathcal{K}_{r_u}\} Q_u,\ \forall k, u, \tag{9b}$$

$$a_n(k) \in \{0,1\},\ b_{n,m}(k) \in \{0,1\},\ \forall k, n, m, \tag{9c}$$

$$\|\mathbf{w}_n(k)\|^2 \leq P^{\max},\ \forall k, n, \tag{9d}$$

$$(1), (2)$$

where (9b) ensures that the downloading rate of the requested PB remains above the user QoS threshold for all potential channel estimation errors. (9c) indicates that the caching and migration variables are binary. In (9d), the transmission power of each edge node is restricted by power budget $P^{\max}$. (1) and (2) specifies edge storage capacity and PB migration constraints, respectively.

However, **P1** is an intractable MINLP problem exacerbated by CSI uncertainty, where the number of possible channel errors is infinite. Besides, the centralized control of PB caching and migration results in an extensive decision space as well as significant information exchange overhead, which imposes substantial computational complexity. To efficiently address **P1**, we develop an MADRL-based approach in Section III to enable collaborative decision-making among edge nodes in a distributed manner.

III. Fine-Grained AI Model Caching and Downloading Solution

In this section, we present the proposed solution for FGAMCD. First, we analyze the limitations of existing MADRL methods in addressing **P1**, and provide an overview of our designed multi-agent action semantics network with data augmentation (MAASN-DA) framework, along with its specialized enhancements. Next, we detail the individual components of MAASN-DA framework, followed by the development of the overall training algorithm.

A. Overview of MAASN-DA Framework

In **P1**, each edge node serves as an agent responsible for determining its caching, migration, and beamforming decisions. Meanwhile, the decisions made by different agents have mutual influence, and jointly affect the model downloading delay, which align with the MADRL framework [32]–[34]. For instance, [34] exploited multi-agent deep deterministic policy gradient (MADDPG) to optimize model partitioning, tolerance latency, and updating frequency for split federated learning, thereby coordinating the global model convergence across multiple edge nodes. However, none of the existing MADRL methods can be directly applied to solve **P1** due to the following reasons: *1) Implicitly Characterization of Actions' Mutual Influence:* Existing methods generally employ a single black-box neural network to output each agent's action, neglecting the fact that different action dimensions may have distinct impacts on other agents, which is inefficient for capturing the complex coupling among the caching and migration policies of multiple edge nodes. *2) Low Sample Efficiency:* MADRL methods suffer from inefficient sample collection due to the costly agent-environment interactions, which can significantly impede the training speed and hinder the model convergence. *3) Multi-Agent Credit Assignment:* MADRL methods often rely on a centralized training stage that leverages a global critic function to learn distributed policies, which leads to challenges in fairly evaluating the contributions of individual agents [35].

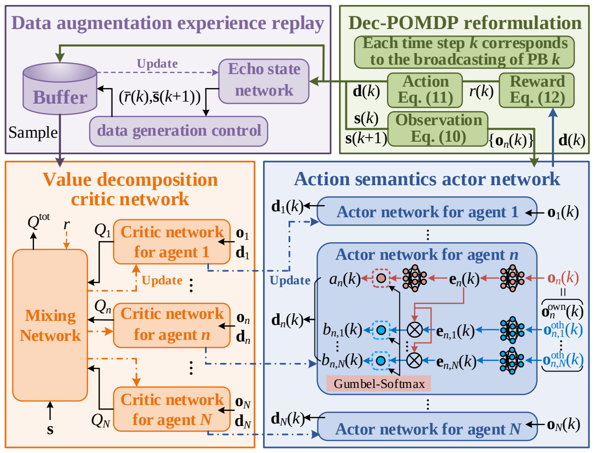


Fig. 3. The proposed MAASN-DA framework.

To tackle the above limitations, we propose MAASN-DA framework as depicted in Fig. 3, which incorporates four main components:

- **Dec-POMDP Reformulation:** The original **P1** is first reformulated as a decentralized partially observable Markov decision process (Dec-POMDP), specifically designed to enable MADRL agents to make sequential decisions for PB caching and delivery.
- **Action Semantics Actor Network:** A specialized actor network is designed for each agent to explicitly capture the effects of different action dimensions on other agents (i.e., action semantics). This network consists of several sub-modules, each processing distinct parts of the agent's observation and producing corresponding action dimensions based on the action semantics.
- **Data Augmentation Experience Replay:** After collecting experience tuples through interactions with the environment, we utilize an echo state network (ESN) to generate additional training samples by leveraging its prediction capabilities, thereby enriching the replay buffer and enhancing sample efficiency. Furthermore, both the quality and quantity of the generated samples are carefully controlled to ensure the convergence performance.
- **Value Decomposition Critic Network:** Each agent employs a local critic to estimate its individual Q-value, which is subsequently combined to obtain the global Q-value via a mixing network. During the training phase, the global Q-value is decomposed to compute the policy

gradients of individual agents, thus addressing the credit assignment problem.

We present detailed elucidation of these components in the following subsections.

B. Dec-POMDP Reformulation

Dec-POMDP features the distributed decision-makings for multiple agents (i.e., edge nodes) with partially observations, which can be defined by $(\mathcal{O},\mathcal{S},\mathcal{D},\pi,r,\gamma)$, where $\mathcal{O}$, $\mathcal{S}$, and $\mathcal{D}$ are observation, state, and action space, respectively. In this work, we optimize the caching and downloading of each PB sequentially, so the index of PB k corresponds to the time step in MADRL. In an arbitrary step k, each agent n obtains an observation $\mathbf{o}_n(k) \in \mathcal{O}$, which is a part of global state $\mathbf{s}(k) \in \mathcal{S}$, then takes action $\mathbf{d}_n(k) \in \mathcal{D}$ based on distributed policy π. Given reward function $r(k)$ and discount factor γ, all agents collaborate to learn an optimal π that maximizes the accumulative discounted reward. We specify these elements in the sequel.

1) Observation: $\mathbf{o}_n(k)$ can be intuitively divided into two parts: $\mathbf{o}_n^{\text{own}}(k)$ that contains environmental information and agent n's own properties, as well as $\mathbf{o}_{n,m}^{\text{oth}}(k), m \in \mathcal{N}\setminus\{n\}$ that represents the observations of agent n on other agents. They are written as

$$\mathbf{o}_n^{\text{own}}(k) = \left[S(k), \{\mathbb{I}\{k \in \mathcal{K}_{r_u}\} : u \in \mathcal{U}_n\}, \tilde{C}_n(k)\right], \quad (10a)$$

$$\mathbf{o}_{n,m}^{\text{oth}}(k) = \varpi_{n,m} \times \left[R_{n,m}^{\text{bac}}(k), \{\mathbb{I}\{k \in \mathcal{K}_{r_u}\} : u \in \mathcal{U}_m\}, \tilde{C}_m(k)\right], \quad (10b)$$

$$\mathbf{o}_n(k) = \left[\mathbf{o}_n^{\text{own}}(k), \left\{\mathbf{o}_{n,m}^{\text{oth}}(k) : m \in \mathcal{N}\setminus\{n\}\right\}\right], \quad (10c)$$

where $\mathcal{U}_n$ denotes the set of users associated with node n. $\tilde{C}_n(k)$ signifies the remaining storage capacity before caching PB k with $\tilde{C}_n(k+1) = \{\tilde{C}_n(k) - a_n(k) S(k)\}^+$ and $\tilde{C}_n(1) = C_n$. Binary indicator $\varpi_{n,m} = 1$ implies that the information of node m can be observed by n, otherwise $\varpi_{n,m} = 0$. Note that $\varpi_{n,m}$ is determined by the distribution of edge nodes and the allowed information exchange overhead. Moreover, the global state in step k is $\mathbf{s}(k) = [\mathbf{o}_n(k) : n \in \mathcal{N}]$.

2) Action: Theoretically, all optimization variables of **P1** $\mathbf{a}, \mathbf{b}, \mathbf{w}$ can be regarded as the agents' actions. Nonetheless, the optimality of MADRL degrades and the training complexity increases with the growth of action dimension. As a remedy, we propose to decouple the variables into two groups, where the PB caching and migration are optimized by MADRL, i.e., the action of agent n in step k is

$$\mathbf{d}_n(k) = [a_n(k), \{b_{n,m}(k) : m \in \mathcal{N}\setminus\{n\}\}], \quad (11)$$

then the high-dimensional beamforming $[\mathbf{w}_n(k) : \forall n]$ is derived from an optimization subroutine with given joint action $\mathbf{d}(k) = [\mathbf{d}_n(k) : \forall n]$, which will be detailed in Section III-F. Accordingly, the distributed policy that maps local observation to action is expressed as $\mathbf{d}_n(k) = \pi(\mathbf{o}_n(k))$.

3) Reward: According to **P1**, the reward design targets to minimize the downloading delay $T(k)$ of each PB k while meeting the constraints, i.e.,

$$r(k) = \begin{cases} -T(k) - r_1 \Lambda(k), \\ \qquad \sum_{u\in\mathcal{U}} \mathbb{I}\{k \in \mathcal{K}_{r_u}\} > 0, \sum_{n\in\mathcal{N}} \lambda_n(k) > 0, \\ -r_2, \ \sum_{u\in\mathcal{U}} \mathbb{I}\{k \in \mathcal{K}_{r_u}\} > 0, \sum_{n\in\mathcal{N}} \lambda_n(k) = 0, \\ 0, \ \text{otherwise}, \end{cases} \quad (12)$$

where $r_1, r_2 > 0$ denote penalties for violating the constraints. To be specific, $\Lambda(k) = 1$ if the optimization subroutine returns an infeasible solution for QoS or transmission power constraints, and all agents receive penalty r_1, otherwise $\Lambda(k) = 0$. If PB k is requested by some users whereas no agent can deliver PB k, then agents are punished[8] by r_2. If PB k is not required by any users, we set $r(k) = 0$. Note that the other constrains in **P1** can be ensured by our actor network design.

C. Action Semantics Actor Network

It is observed from (11) that $a_n(k)$ only affects the agent n's own property, i.e., the remaining storage capacity, while the PB migration $b_{n,m}(k)$ directly impacts the caching and downloading of other agents. This motivates us to explicitly extract the action semantics by dividing the black-box actor network into multiple sub-modules, each processing a distinct part of the agent's observation [36]. As shown in Fig. 3, the actor network of each agent n consists of N sub-modules. The first one takes the full observation $\mathbf{o}_n(k)$ as input, utilizes two neural networks to yield observation embedding $\mathbf{e}_n(k)$ and $a_n(k)$, respectively. The rest $N-1$ sub-modules contain $N-1$ neural networks to generate the action dimensions related with other influenced agents. Each of the $N-1$ sub-modules (say the m-th sub-module) uses a part of observation $\mathbf{o}_{n,m}^{\text{oth}}(k)$ as input to determine embedding $\mathbf{e}_{n,m}(k)$ related with influenced agent m, then combines $\mathbf{e}_n(k)$ and $\mathbf{e}_{n,m}(k)$ via inner product to output $b_{n,m}(k)$.

Following the above structure, the action semantics actor network of agent n is represented by $\pi(\mathbf{o}_n(k); \varphi_n)$ with φ_n being the network parameters. For the sake of satisfying constraints (1), (2), and (9c), we further adapt the output layer of $\pi(\mathbf{o}_n(k); \varphi_n)$ as below. Given an output value $\tilde{d}$, we leverage the Gumbel-Softmax reparameterization to yield differentiable binary variable as

$$d = GS\left(\tilde{d}\right) = \text{Sigmoid}\left(\frac{\tilde{d} + \ln\varsigma - \ln(1-\varsigma)}{s_{\text{temp}}}\right), \quad (13)$$

where $\text{Sigmoid}(x) = \frac{e^x}{1+e^x}$, ς is a random variable obeying uniform distribution within [0, 1], s_{temp} denotes the Softmax coefficient, and smaller s_{temp} makes d closer to a binary variable. Therefore, given the variables output by the sub-modules of $\pi(\mathbf{o}_n(k); \varphi_n)$, denoted by $\tilde{a}_n(k)$ and $\tilde{b}_{n,m}(k)$, we invoke the following operations:

$$a_n(k) = GS(\tilde{a}_n(k)), \ b_{n,m}(k) = GS(\tilde{b}_{n,m}(k)), \ \forall m, \quad (14)$$

[8]Two factors may lead to insufficient edge storage capacity for caching a PB requested by certain users. First, the caching decisions may fail to fully utilize the available storage, in which case the reward penalty guides the agents to adjust their policies. Second, the total system resources may be inadequate to satisfy all user requests, rendering problem **P1** mathematically infeasible. In such situations, users can either defer to the next request period or directly retrieve the models from the cloud.

In addition, if the edge node's storage capacity is not adequate for caching PB k, i.e., $\tilde{C}_n(k) < S(k)$, $a_n(k)$ is forced to be 0. For $a_n(k) = 0$, we set $b_{n,m}(k) = 0, m \in \mathcal{N} \backslash \{n\}$ to prohibit PB migration from node n.[9]

D. Data Augmentation Experience Replay

In each time step k, all agents interact with the environment and collect a new experience tuple $(\mathbf{s}(k), \mathbf{d}(k), r(k), \mathbf{s}(k+1))$. In the meantime, we adopt an ESN to predict the reward and next state $(\bar{r}(k), \bar{\mathbf{s}}(k+1))$, thereby synthesizing training data and boosting sample efficiency. The reason for choosing an ESN over other predictive models, such as recurrent neural networks (RNNs) or transformers, is that it only requires training the output weights while keeping the high-dimensional hidden weights fixed. This significantly reduces training complexity and mitigate gradient vanishing/exploding issues.

Specifically, ESN consists of input layer, reservoir layer, and output layer with weighting parameters $\boldsymbol{\eta}_{\text{in}}$, $\boldsymbol{\eta}_{\text{re}}$, and $\boldsymbol{\eta}_{\text{out}}$, respectively. Taking sequence $\mathbf{v}(1), \ldots, \mathbf{v}(k)$ as input, where $\mathbf{v}(k) = (\mathbf{s}(k), \mathbf{d}(k))$, ESN recurrently updates the reservoir states $\mathbf{q}(1), \ldots, \mathbf{q}(k)$ and yields the prediction output as follows [37]:

$$\begin{aligned} &\mathbf{q}(k) = \tanh\left(\boldsymbol{\eta}_{\text{in}}\mathbf{v}(k) + \boldsymbol{\eta}_{\text{re}}\mathbf{q}(k-1)\right), \\ &(\bar{r}(k), \bar{\mathbf{s}}(k+1)) = \boldsymbol{\eta}_{\text{out}}\mathbf{q}(k). \end{aligned} \tag{15}$$

We fix $\boldsymbol{\eta}^{\text{in}}$ and $\boldsymbol{\eta}^{\text{re}}$ after random initialization, and only update $\boldsymbol{\eta}^{\text{out}}$ during the tuning of ESN. The loss function to be minimized is given by

$$L^{\text{e}}(\boldsymbol{\eta}_{\text{out}}) = \left\|\boldsymbol{\eta}_{\text{out}}\mathbf{q}(k) - (r(k), \mathbf{s}(k+1))\right\|^2, \tag{16}$$

where real experience tuples from the environment $(r(k), \mathbf{s}(k+1)), \forall k$ serve as labels for tuning the ESN.

Considering that the extensive inclusion of synthetic training samples may degrade the convergence performance, we design a data generation control strategy to filter out low-quality data while adaptively adjusting the number of synthetic samples. Concretely, we invoke the following condition to identify satisfactory data predicted by ESN:

$$\left\|(\bar{r}(k), \bar{\mathbf{s}}(k+1)) - (r(k), \mathbf{s}(k+1))\right\| \le \xi, \tag{17}$$

where ξ denotes the data selection threshold. Moreover, the ESN can generate at most $\tau_0 K$ synthetic samples per episode in the initial training stage with $\tau_0 \in [0,1]$ being a tunable proportion. The number of synthetic samples should decrease with the training episodes, thus we have

$$\tau_e = \left\lfloor \tau_0 K \Lambda^{\lfloor e/\bar{E} \rfloor} \right\rfloor, \tag{18}$$

where τ_e represents the maximum number of synthetic samples in the e-th episode, $\Lambda \in (0,1)$ is the attenuation factor, and the number of synthetic samples declines every $\bar{E}$ episodes. Thereafter, both real experience tuples and synthetic samples are stored in the replay buffer.

[9]It is worth noting that an edge node with fully occupied storage can still relay PBs received from other nodes to users. This is because such PBs are only temporarily buffered in volatile memory (e.g., DRAM/VRAM) rather than persistently cached, and thus do not consume the node's storage capacity.

E. Value Decomposition Critic Network

The critic network of MAASN-DA is designed based on the value decomposition architecture, allowing for the evaluation of each agent's contribution to the overall system. Each agent n possesses a local critic $Q(\mathbf{o}_n(k), \mathbf{d}_n(k); \boldsymbol{\theta}_n)$ with parameters $\boldsymbol{\theta}_n$ to estimate its Q-value Q_n, then a mixing network is adopted to combine the individual Q-value of each agent, and produce a global Q-value as

$$Q^{\text{tot}} = g^{\text{mix}}\left(\mathbf{s}, Q_1, \ldots, Q_N; \boldsymbol{\theta}^{\text{mix}}\right), \tag{19}$$

where g^{mix} indicates the mixing function, and $\boldsymbol{\theta}^{\text{mix}}$ is the parameters of the mixing network. Inspired by the well-known QMIX algorithm [38], we design g^{mix} by forcing monotonicity between Q^{tot} and each Q_n, i.e.,

$$\partial Q^{\text{tot}} / \partial Q_n \ge 0, \ \forall n \in \mathcal{N}, \tag{20}$$

such that any action $\mathbf{d}_n(k)$ that increases the individual Q-value Q_n also improves the global Q-value Q^{tot}. To this end, the parameter $\boldsymbol{\theta}^{\text{mix}}$ is yielded by separate hypernetworks, each of which employs an absolute activation function to output the weights for one layer of the mixing network, thereby guaranteeing the nonnegativity.

During the training stage, we randomly sample a mini-batch of samples in the replay buffer, and perform forward propagation through the neural networks in MAASN-DA to compute the loss functions. To be specific, the loss for the value decomposition critic network is given by

$$L^{\text{c}}\left(\{\boldsymbol{\theta}_n\}, \boldsymbol{\theta}^{\text{mix}}\right) = \mathbb{E}\left[Q^{\text{tar}}(k) - Q^{\text{tot}}\left(\mathbf{s}(k), \mathbf{d}(k); \{\boldsymbol{\theta}_n\}, \boldsymbol{\theta}^{\text{mix}}\right)\right], \tag{21}$$

where

$$Q^{\text{tar}}(k) = r(k) + \gamma Q^{\text{tot}}\left(\mathbf{s}(k+1), \{\pi(\mathbf{o}_n(k+1); \hat{\boldsymbol{\varphi}}_n)\}; \{\hat{\boldsymbol{\theta}}_n\}, \hat{\boldsymbol{\theta}}^{\text{mix}}\right),$$

$\hat{\boldsymbol{\varphi}}_n$ and $\hat{\boldsymbol{\theta}}_n, \hat{\boldsymbol{\theta}}^{\text{mix}}$ are the parameters of the target actor and critic networks, respectively. These target networks share the same structure as the main actor and critic networks, while their parameters are slowly updated to stabilize training. Additionally, the global Q-value is decomposed to evaluate the contribution of each agent n, thereby calculating its actor loss as follows:

$$L^{\text{a}}(\boldsymbol{\varphi}_n) = \mathbb{E}\left[-Q\left(\mathbf{o}_n(k), \pi(\mathbf{o}_n(k); \boldsymbol{\varphi}_n); \boldsymbol{\theta}_n\right)\right], \ \forall n. \tag{22}$$

Consequently, the actor and critic networks of MAASN-DA are updated using gradient descent based on the above loss functions.

F. Robust CoMP Beamforming Subroutine

To proceed with, we propose a robust CoMP beamforming subroutine to optimize $[\mathbf{w}_n(k) : \forall n]$ under CSI uncertainty. It is notable that with given PB caching and migration decisions $\mathbf{d}(k)$, the participation of each edge node, i.e., $\lambda_n(k)$, can be determined by (3). We define $\mathbf{w}(k) = \left[\mathbf{w}_1^{\text{T}}(k), \ldots, \mathbf{w}_N^{\text{T}}(k)\right]^{\text{T}} \in \mathbb{C}^{MN \times 1}$ and $\mathbf{W}(k) = \mathbf{w}(k)\mathbf{w}^{\text{H}}(k) \in \mathbb{C}^{MN \times MN}$, then the beamforming problem for broadcasting PB k can be recast as

$$\mathbf{P2}(k) : \min_{\mathbf{W}(k), \zeta} \frac{1}{\zeta}, \tag{23a}$$

$$\text{s.t.} \min_{\mathbf{e}_{n,u}(k)\in\mathcal{E}_{n,u},\forall n} R_u(k) \geq \mathbb{I}\{k\in\mathcal{K}_{r_u}\}Q_u,\ \forall u, \tag{23b}$$

$$\sum\nolimits_{l=(n-1)M+1}^{nM}[\mathbf{W}(k)]_{l,l}\leq P^{\max},\ \forall n, \tag{23c}$$

$$\frac{1}{\zeta}\geq\frac{\mathbb{I}\{k\in\mathcal{K}_{r_u}\}S(k)}{\min\limits_{\mathbf{e}_{n,u}(k)\in\mathcal{E}_{n,u},\forall n}R_u(k)},\ \forall u, \tag{23d}$$

$$\mathbf{W}(k)\succeq 0, \tag{23e}$$

$$\operatorname{rank}[\mathbf{W}(k)]=1, \tag{23f}$$

where ζ is an auxiliary variable with $\frac{1}{\zeta}$ being the maximum worst case broadcasting delay among users, and $R_u(k)$ can be rewritten as

$$R_u(k)=B\times$$
$$\log_2\!\left(1+\left[\tilde{\mathbf{h}}_u(k)+\mathbf{e}_u(k)\right]^{\mathrm{H}}\mathbf{W}(k)\left[\tilde{\mathbf{h}}_u(k)+\mathbf{e}_u(k)\right]/\sigma_u^2\right), \tag{24}$$

where $\tilde{\mathbf{h}}_u(k)=\left[\lambda_1(k)\tilde{\mathbf{h}}_{1,u}^{\mathrm{T}}(k),\ldots,\lambda_N(k)\tilde{\mathbf{h}}_{N,u}^{\mathrm{T}}(k)\right]^{\mathrm{T}}$ and $\mathbf{e}_u(k)=\left[\lambda_1(k)\mathbf{e}_{1,u}^{\mathrm{T}}(k),\ldots,\lambda_N(k)\mathbf{e}_{N,u}^{\mathrm{T}}(k)\right]^{\mathrm{T}}$ are the stacked imperfect channel and estimation error for user u, respectively. To handle the infinite possibilities of channel errors in constraints (23b) and (23d), we invoke S-Procedure as illustrated in the following lemma [39].

Lemma 1: Let $\mathbf{A}_1,\mathbf{A}_2\in\mathbb{H}^{MN\times MN}$, $\mathbf{b}_1,\mathbf{b}_2\in\mathbb{C}^{MN\times1}$, $\kappa_1,\kappa_2\in\mathbb{R}$, for $\forall\mathbf{e}\in\mathbb{C}^{MN\times1}$, the implication

$$\mathbf{e}^{\mathrm{H}}\mathbf{A}_1\mathbf{e}+2\operatorname{Re}\left\{\mathbf{b}_1^{\mathrm{H}}\mathbf{e}\right\}+\kappa_1\leq0 \tag{25a}$$
$$\Rightarrow\mathbf{e}^{\mathrm{H}}\mathbf{A}_2\mathbf{e}+2\operatorname{Re}\left\{\mathbf{b}_2^{\mathrm{H}}\mathbf{e}\right\}+\kappa_2\leq0 \tag{25b}$$

holds if and only if there exists a $\varepsilon\geq0$ such that

$$\begin{bmatrix}\varepsilon\mathbf{A}_1-\mathbf{A}_2 & \varepsilon\mathbf{b}_1-\mathbf{b}_2\\ \varepsilon\mathbf{b}_1^{\mathrm{H}}-\mathbf{b}_2^{\mathrm{H}} & \varepsilon\kappa_1-\kappa_2\end{bmatrix}\succeq\mathbf{0}. \tag{26}$$

Take (23b) as an example, we first rewrite it as

$$\max_{\mathbf{e}_{n,u}(k)\in\mathcal{E}_{n,u},\forall n}\mathbb{I}\{k\in\mathcal{K}_{r_u}\}\cdot\Big[-\mathbf{e}_u^{\mathrm{H}}(k)\mathbf{W}(k)\mathbf{e}_u(k)$$
$$+2\operatorname{Re}\left\{-\tilde{\mathbf{h}}_u^{\mathrm{H}}(k)\mathbf{W}(k)\mathbf{e}_u(k)\right\}+\kappa_u^{\langle1\rangle}(k)\Big]\leq0,\ \forall u, \tag{27}$$

where $\kappa_u^{\langle1\rangle}(k)=\sigma_u^2\left(2^{Q_u/B}-1\right)-\tilde{\mathbf{h}}_u^{\mathrm{H}}(k)\mathbf{W}(k)\tilde{\mathbf{h}}_u(k)$. According to (4), we also have

$$\mathbf{e}_u^{\mathrm{H}}(k)\mathbf{C}_u\mathbf{e}_u(k)-N\leq0,\ \mathbf{C}_u=\begin{bmatrix}\mathbf{C}_{1,u}\cdots & \mathbf{O}\\ \vdots\ \ddots & \vdots\\ \mathbf{O} & \cdots\mathbf{C}_{N,u}\end{bmatrix},\forall u. \tag{28}$$

Then, by substituting (28) into (25a) and (27) into (25b), constraint (23b) can be transformed into

$$\mathbb{I}\{k\in\mathcal{K}_{r_u}\}\cdot\begin{bmatrix}\varepsilon_u^{\langle1\rangle}\mathbf{C}_u+\mathbf{W}(k) & \mathbf{W}^{\mathrm{H}}(k)\tilde{\mathbf{h}}_u(k)\\ \tilde{\mathbf{h}}_u^{\mathrm{H}}(k)\mathbf{W}(k) & -\varepsilon_u^{\langle1\rangle}N-\kappa_u^{\langle1\rangle}(k)\end{bmatrix}\succeq\mathbf{0},\forall u, \tag{29}$$

where $\varepsilon_u^{\langle1\rangle}$ is a non-negative auxiliary variable. Analogously, we convert (23d) into

$$\mathbb{I}\{k\in\mathcal{K}_{r_u}\}\cdot\begin{bmatrix}\varepsilon_u^{\langle2\rangle}\mathbf{C}_u+\mathbf{W}(k) & \mathbf{W}^{\mathrm{H}}(k)\tilde{\mathbf{h}}_u(k)\\ \tilde{\mathbf{h}}_u^{\mathrm{H}}(k)\mathbf{W}(k) & -\varepsilon_u^{\langle2\rangle}N-\kappa_u^{\langle2\rangle}(k)\end{bmatrix}\succeq\mathbf{0},\forall u, \tag{30}$$

where $\kappa_u^{\langle2\rangle}(k)=\sigma_u^2\left(2^{\zeta S(k)/B}-1\right)-\tilde{\mathbf{h}}_u^{\mathrm{H}}(k)\mathbf{W}(k)\tilde{\mathbf{h}}_u(k)$.

Subsequently, we tackle non-convex rank constraint (23f) by constructing an equivalent difference-of-convex (DC) expression. Specifically, $\operatorname{rank}[\mathbf{W}(k)]=1$ if and only if $\mathbf{W}(k)$ possesses only one nonzero eigenvalue, i.e., $\sigma_1[\mathbf{W}(k)]>0,\sigma_m[\mathbf{W}(k)]=0,m\in\{2,\ldots,MN\}$, thus we have

$$\operatorname{tr}[\mathbf{W}(k)]=\sum_{m=1}^{MN}\sigma_m[\mathbf{W}(k)]=\sigma_1[\mathbf{W}(k)]=\|\mathbf{W}(k)\|. \tag{31}$$

As a consequence, define $\boldsymbol{\varepsilon}=\left[\varepsilon_u^{\langle1\rangle},\varepsilon_u^{\langle2\rangle}:\forall u\right]$, we can transform **P2** into the following DC program:

$$\textbf{P2.1}(k):\min_{\mathbf{W}(k),\zeta,\boldsymbol{\varepsilon}}\frac{1}{\zeta}+\mu\left(\operatorname{tr}[\mathbf{W}(k)]-\|\mathbf{W}(k)\|\right), \tag{32}$$
$$\text{s.t. (23c), (23e), (29), (30)}$$

where μ is the regularization parameter [40]. Then, we linearize the concave part $-\|\mathbf{W}(k)\|$ in (32) and reach a stationary solution by iteratively solving **P2.2**(k) in the sequel

$$\textbf{P2.2}(k):\min_{\mathbf{W}(k),\zeta,\boldsymbol{\varepsilon}}\frac{1}{\zeta}+\mu\left(\operatorname{tr}[\mathbf{W}(k)]-\operatorname{tr}\left[\bar{\mathbf{u}}_1\bar{\mathbf{u}}_1^{\mathrm{H}}\mathbf{W}(k)\right]\right), \tag{33}$$
$$\text{s.t. (23c), (23e), (29), (30)}$$

where $\bar{\mathbf{u}}_1\bar{\mathbf{u}}_1^{\mathrm{H}}=\frac{\partial\|\mathbf{W}\|}{\partial\mathbf{W}}\Big|_{\mathbf{W}=\bar{\mathbf{W}}(k)}$ with $\bar{\mathbf{u}}_1$ denoting the eigenvector corresponding to the maximum eigenvalue of $\bar{\mathbf{W}}(k)$, and $\bar{\mathbf{W}}(k)$ is the solution obtained from the previous iteration. Notice that **P2.2**(k) pertains to convex optimization issue that can be solved by interior-point method (IPM). Finally, the proposed robust CoMP beamforming subroutine converges to a rank-one solution $\mathbf{W}(k)$, then we reconstruct beamforming vector $\mathbf{w}(k)$ through eigenvalue decomposition.

Remark 2 (Solution Quality of the Beamforming Subroutine): The proposed approach leverages the S-Procedure to handle channel uncertainty and reformulates the rank-1 constraint via DC programming. The introduction of auxiliary variables, each with a properly defined feasible region according to S-Procedure theory, does not affect optimality [39]. The DC program is iteratively solved using a first-order approximation of the spectral norm, which may lead to sub-optimal solutions. Nevertheless, as reported in [40], this approach achieves superior solution quality and computational efficiency compared to the conventional Gaussian randomization method for recovering rank-1 solutions.

G. Overall Training Algorithm for MAASN-DA

The developed training algorithm for MAASN-DA is outlined in **Algorithm 1**. In each time step k, the edge node agents make PB caching and migration decisions exploiting the action semantics actor network, then the CoMP beamforming is derived from the embedded optimization subroutine (Lines 4-9). After collecting experience tuples via agent-environment interactions, we invoke the ESN to synthesize additional training samples while tuning the ESN, followed by performing data generation control (Lines 10-19). Afterwards, the global Q-value is obtained based on the value decomposition critic network, then the actor and critic network parameters are updated by minimizing the loss functions (Lines 20-24). Finally, the well-trained agents can efficiently coordinate AI

Algorithm 1 Overall Training Algorithm for MAASN-DA

```
1: Input: AI model repository $J, K, \mathcal{K}_j, \forall j$, user requests $U, r_u, Q_u, \forall u$, FGAMCD environment $N, M, B, P^{\max}, C_n, \tilde{\mathbf{h}}_{n,u}(k), \mathbf{C}_{n,u}, R^{\mathrm{bac}}_{n,m}(k), \forall n, u, m$, and learning hyperparameters.
2: Initialize: Parameter of ESN $\boldsymbol{\eta}^{\mathrm{in}}, \boldsymbol{\eta}^{\mathrm{re}}, \boldsymbol{\eta}^{\mathrm{out}}$, actor network $\boldsymbol{\varphi}_n, \forall n$, critic network $\boldsymbol{\theta}_n, \boldsymbol{\theta}^{\mathrm{mix}}, \forall n$, and target networks $\hat{\boldsymbol{\varphi}}_n, \hat{\boldsymbol{\theta}}_n, \hat{\boldsymbol{\theta}}^{\mathrm{mix}}, \forall n$, replay buffer $\mathcal{B}$.
3: for $e \in \mathcal{E} = \{1, \dots, E\}$ do
4:     for $k \in \mathcal{K}$ do
5:         Each agent $n$ observes $\mathbf{o}_n(k)$ and takes action $\mathbf{d}_n(k) = \pi(\mathbf{o}_n(k); \boldsymbol{\varphi}_n)$.
6:         Iteratively solve P2.2$(k)$ to optimize CoMP beamforming $\mathbf{w}(k)$.
7:         Calculate reward $r(k)$ and transit to the next state $\mathbf{s}(k+1)$.
8:         Store $(\mathbf{s}(k), \mathbf{d}(k), r(k), \mathbf{s}(k+1))$ in $\mathcal{B}$.
9:     end for
10:    while 1 do
11:        Sample experience $(\mathbf{s}(k), \mathbf{d}(k), r(k), \mathbf{s}(k+1))$ from $\mathcal{B}$.
12:        Invoke the ESN to predict $(\bar{r}(k), \bar{\mathbf{s}}(k+1))$.
13:        Tune the ESN by minimizing $L^{\mathrm{e}}(\boldsymbol{\eta}_{\mathrm{out}})$ in (16).
14:        if $\|(\bar{r}(k), \bar{\mathbf{s}}(k+1)) - (r(k), \mathbf{s}(k+1))\| \le \xi$ then
15:            Store $(\mathbf{s}(k), \mathbf{d}(k), \bar{r}(k), \bar{\mathbf{s}}(k+1))$ in $\mathcal{B}$.
16:        else if the number of synthetic samples reaches $\tau_e$ then
17:            break
18:        end if
19:    end while
20:    Sample a mini-batch of samples from $\mathcal{B}$.
21:    Calculate the global Q-value using (19).
22:    Update the critic network by minimizing $L^{\mathrm{c}}(\{\boldsymbol{\theta}_n\}, \boldsymbol{\theta}^{\mathrm{mix}})$ in (21).
23:    Update the actor network by minimizing $L^{\mathrm{a}}(\boldsymbol{\varphi}_n), \forall n$ in (22).
24:    Slowly update the target networks.
25: end for
26: Output: Well-trained edge node agents for optimizing $\mathbf{a}, \mathbf{b}, \mathbf{w}$.
```

model caching and delivery processes, responding to diverse user requests and uncertain channel estimations, thus ensuring low-latency AI model downloading.

IV. Theoretical Analysis for MAASN-DA

In this section, we present theoretical analysis for the proposed MAASN-DA approach. To begin with, we analyze the convergence performance of critic network, characterizing the approximation error of Q-value and offering guidelines for configuring the hyperparameters of data augmentation experience replay. Afterwards, the convergence of actor network is analyzed, ensuring the achievement of local-optimal PB caching and downloading decisions. Lastly, we present complexity analysis for MAASN-DA.

A. Convergence Analysis for Critic Network

We commence by making some assumptions that are commonly utilized in neural network analysis [10].

Assumption 1: The state, action, and reward in each time step are bounded by $\|\mathbf{s}(k)\| \le B_{\mathbf{s}}, \|\mathbf{d}(k)\| \le B_{\mathbf{d}}, |r(k)| \le B_r, \forall k$. This assumption is typically met due to all elements in $\mathbf{s}(k)$, $\mathbf{d}(k)$ and $r(k)$ possess explicit physical meanings.

Assumption 2: The ESN for generating synthetic samples satisfies the echo state property, i.e., the spectral norms of weight matrices satisfy $\|\boldsymbol{\eta}_{\mathrm{in}}\| \le \psi^{\max}_{\mathrm{in}}$, $\|\boldsymbol{\eta}_{\mathrm{re}}\| \le \psi^{\max}_{\mathrm{re}} < 1$, and $\boldsymbol{\eta}_{\mathrm{out}} \le \psi^{\max}_{\mathrm{out}}$. This assumption can be met by reasonably initializing the weights of the ESN [37].

Assumption 3: The value decomposition critic network can be fitted by a deep recurrent Q network (DRQN) with input, recurrent, and output weights $\boldsymbol{\omega}_{\mathrm{in}}$, $\boldsymbol{\omega}_{\mathrm{re}}$, and $\boldsymbol{\omega}_{\mathrm{out}}$, respectively. This RNN is able to estimate the Q-value as: $\mathbf{p}(k) = \tanh(\boldsymbol{\omega}_{\mathrm{in}}\mathbf{v}(k) + \boldsymbol{\omega}_{\mathrm{re}}\mathbf{p}(k-1)), Q(\mathbf{s}(k), \mathbf{d}(k)) = \boldsymbol{\omega}_{\mathrm{out}}\mathbf{p}(k)$, with given the state sequence up to step k. This assumption stems from the wide adaptation of DRQN in value decomposition MADRL [38].

Assumption 4: The training of the critic network can be simplified as the fitted-Q iteration algorithm [41], in which the Q-value function is updated E episodes, and the experience relay is deemed as a sampling distribution Ω over $\mathcal{S} \times \mathcal{D}$. Since the actor network is updated to maximize the Q-value estimated by the critic, this yields the greedy policy π_E after E iterations.

Under Assumption 1-4, we establish the following theorem to illustrate the convergence of the critic network.

Theorem 1: Let Q_{π_E} be the Q-value corresponding to π_E and $Q^* = \sup_\pi Q_\pi$ be the optimal Q-value. The Q-value approximation error is upper-bounded by

$$\mathbb{E}_{\Xi}\left[|Q_{\pi_E} - Q^*|\right] \le \underbrace{\frac{4\gamma^{E+1}}{(1-\gamma)^2}B_r}_{\text{Algorithmic error}}$$

$$+ \underbrace{\vartheta_{\Xi,\Omega}\left[\frac{2\gamma}{(1-\gamma)^2}\sqrt{\nu^{\max}} + \xi\left(1 + \gamma\phi^{\max}_{\mathrm{out}}\phi^{\max}_{\mathrm{in}}\right)\right]}_{\text{Statistical error}}, \quad (34)$$

where Ξ signifies the distribution corresponding to the Dec-POMDP in Section III-B, and $\vartheta_{\Xi,\Omega}$ denotes the concentration coefficient determined by Ξ and Ω. $\phi^{\max}_{\mathrm{in}}$ and $\phi^{\max}_{\mathrm{out}}$ are the bounds of $\|\boldsymbol{\omega}_{\mathrm{in}}\|$ and $\|\boldsymbol{\omega}_{\mathrm{out}}\|$, respectively. $\nu^{\max}$ represents the maximum one-step error, given by

$$\nu^{\max} \le \underbrace{4\max\left\{B_r/(1-\gamma) - \varsigma L^{\mathrm{DRQN}}_{\varsigma}, 0\right\}^2}_{\text{Estimation bias}}$$

$$+ \underbrace{D_1 V^2 \ln U^{\mathrm{DRQN}}_{\varsigma}/K'}_{} + \underbrace{D_2 V^2 \varsigma}_{\text{Estimation variance}}, \quad (35)$$

[10]Please note that these assumptions are only imposed to derive the closed-form convergence bound and extract theoretical insights; the actual implementation of the proposed algorithm does not depend on them. Violating these assumptions may lead to inaccurate bound characterization and sub-optimal hyperparameter selection, thereby degrading training convergence.

where

$$K' = K(1+\tau_0 - \frac{\tau_0}{\xi}\{\psi_{\text{out}}^{\max}\psi_{\text{in}}^{\max}\sqrt{B_{\mathbf{s}}^2+B_{\mathbf{d}}^2}\frac{1-(\psi_{\text{re}}^{\max})^K}{1-\psi_{\text{re}}^{\max}} + \sqrt{B_r^2+B_{\mathbf{s}}^2}\})$$

denotes the number of training samples in each episode, and ς is a positive constant. $D_1 = 8\sqrt{2K'}+256/V$, $D_2 = 4\sqrt{2K'}+52$, $V = B_r/(1-\gamma)$. $L_\varsigma^{\text{DRQN}}$ and $U_\varsigma^{\text{DRQN}}$ are the upper- and lower-bounds of the exterior ς-number of the DRQN, respectively.

Proof: Please refer to Appendix A. ■

Theorem 1 reveals that the Q-value approximation error in (34) is composed by an algorithmic error and a statistical error. The algorithmic error asymptotically diminishing to zero as the training proceeds. The statistical error is caused by the bias and variance that arise in estimating the Q-value function using the critic network (characterized by (35)), as well as the prediction inaccuracies of the ESN.

On the other side, K' encompasses both real and synthetic samples. Different from traditional experience replay that only collect K samples in each episode, the proposed data augmentation trick enlarges K' to boost sample efficiency, thus reducing the variance in Q-value estimation. Furthermore, it is evident that K' is influenced by τ_0 and ξ, and the judicious configure of these hyperparameters can mitigate the Q-value approximation error. Consequently, τ_0 and ξ are obtained by numerically minimizing the upper bound in (34) via a two-dimensional search. A practical example will be provided in Section V-C, where we traverse all possible combinations of τ_0 and ξ to find the one achieving the minimum upper bound value. Note that this search is performed only once prior to training, incurring negligible overhead.

B. Convergence Analysis for Actor Network

We establish the following theorem to guarantee the convergence of the actor network, thereby ensuring the attainment of stable decisions of PB caching, migration, and broadcasting beamforming.

Theorem 2: In each time step of **Algorithm 1**, the CoMP beamforming subroutine converges within a finite number of iterations. Moreover, the action semantics actor network converges to a local optimal policy, i.e.,

$$\lim_{E\to\infty} \|\nabla_{\boldsymbol{\varphi}_n} L^{\text{a}}(\boldsymbol{\varphi}_n)\| = 0,\ \forall n. \tag{36}$$

Proof: Please refer to Appendix B. ■

C. Computational Complexity and Signaling Cost Analysis

The computational complexity of **Algorithm 1** is analyzed below. Typically, the training complexity of a L-layer neural network is $\mathcal{O}\left(\sum_{l=1}^{L-1} 2(\Upsilon_l - 1)\Upsilon_{l+1}\right)$, where Υ_l denotes the number of neurons in the l-th layer. In Appendix C, we present the detailed architectures of actor network, critic network, and ESN, which can be substituted into this formula to calculate the corresponding complexities $\mathcal{O}(Co_{\text{act}})$, $\mathcal{O}(Co_{\text{cri}})$, and $\mathcal{O}(Co_{\text{ESN}})$, respectively. Additionally, suppose that the optimization sub-routine needs I iterations to converge, the beamforming design in each step has a complexity of $\mathcal{O}\left(I\left(M^2N_{\text{par}}^2\right)^{3.5}\right)$ [42], where N_{par} denotes the number of participating edge nodes in broadcasting each PB. Note that $N_{\text{par}} \le N$ since the excessive PB migration among edge nodes is prohibited to reduce backhaul link delay. As a consequence, the overall training complexity for K steps and E episodes is

$$\mathcal{O}\left(EK\left[N(Co_{\text{act}}+Co_{\text{cri}})+Co_{\text{ESN}}+I\left(M^2N_{\text{par}}^2\right)^{3.5}\right]\right). \tag{37}$$

After training, the implementation stage merely requires executing the actor network and the optimization subroutine, leading to a computational complexity of $\mathcal{O}(K[N\Upsilon_{\text{act},1}\Upsilon_{\text{act},L} + I\left(M^2N_{\text{par}}^2\right)^{3.5}])$, where $\Upsilon_{\text{act},l}$ is the number of neurons in the l-th layer of the actor. To form the input observations, edge nodes exchange backhaul rates, user requests, and remaining storage capacities, as specified in $\mathbf{o}_{n,m}^{\text{oth}}(k)$ in (10b). The resultant signaling cost is $\mathcal{O}(K\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{N}\setminus\{n\}}\varpi_{n,m}(|\mathcal{U}_m|+2))$.

TABLE II
Simulation Parameters

Parameter	Value	Parameter	Value
N	6	U	30
J	60	K	450
Q_u	[5, 7] Gbps	C_n	1.25 GB
M	20	υ	-30 dB
α	3	σ_u^2	-80 dBm
$\mathbf{C}_{n,u}$	$10^{10}\mathbf{I}_M$	B	400 MHz
$R_{n,m}^{\text{bac}}(k)$	[8, 12] Gbps	$P^{\max}$	43 dBm
r_1, r_2	10, 10	γ	0.95
τ_0, ξ	0.8, 1.12	Λ	0.8
$E, \bar{E}$	600, 10	μ	1

V. Performance Evaluation

A. Simulation Setting and AI Model Repository Construction

In this section, we conduct simulations to evaluate the performance as well as glean insights of our proposed FGAMCD system. We consider a square area of 1 km^2 with 4-8 edge nodes and 18-48 randomly distributed users. The edge-user wireless channel is generated based on (4), and the small-scale fading $\bar{\mathbf{h}}_{n,u}(k)$ follows Rician channel model with factor of 3. The default parameter setting is listed in Table II, we set $\varpi_{n,m}$ by allowing information exchange between edge nodes less than 500 m away, and the neural network architectures of MAASN-DA are given in Appendix C.

The AI model repository is constructed based on 3 basic AI models, i.e., Inception-v3, ResNet-18, and MobileNet-v2, per-trained on ImageNet. We fine-tune these models on the CIFAR-100 dataset to generate 60 task-specific models. In particular, each basic model is fine-tuned on 20 sub-datasets, each corresponding to one of the 20 superclasses in CIFAR-100 [43]. Therefore, the fine-tuned models are responsible for classifying the images within their respective superclass, e.g., the Inception-v3 model for "people" superclass is used to distinguish "baby", "boy", "girl", "man", and "woman". Users request the 60 models in the repository according to a Zipf distribution, where the request probability of model j is given by $j^{-\iota}/\sum_{i=1}^{60} i^{-\iota}$, where distribution parameter $\iota = 0.5$ in default with sensitivity analysis provided later.

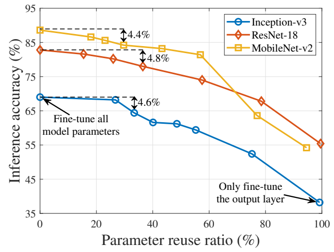


Fig. 4. Inference accuracy versus AI model parameter reuse ratio.

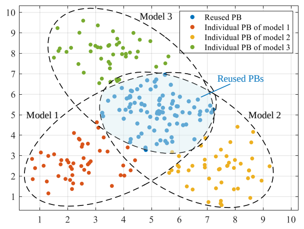


Fig. 5. A visualization of AI model parameter reuse, where different points indicate different PBs.

To capture the parameter reusability, we adopt a two-stage AI model fine-tuning. In the first stage, all parameters of the basic models are fine-tuned on several selected superclasses. In the second stage, we freeze a portion of the parameters, and further fine-tune the models on the remaining superclasses. For instance, all parameters of the basic ResNet-18 are first fine-tuned on "fruit and vegetables" superclass, the resulting model is then fine-tuned on "flower" superclass, with bottom layers frozen during this process. The frozen layers correspond to the reusable PBs in the AI model repository.

B. Effectiveness of FGAMCD

Fig. 4 illustrates the impact of parameter reuse on the inference accuracy of task-specific models, where the parameter reuse ratio is controlled by adjusting the number of frozen layers during the second stage of fine-tuning. As can be observed, when the parameter reuse ratio is moderate, the degradation in inference accuracy is slight compared to full-parameter fine-tuning (i.e., no frozen layer and a reuse ratio of 0). This phenomenon demonstrates that the AI models fine-tuned for different downstream tasks can share a proportion of PBs, validating the effectiveness of our proposed FGAMCD. In the subsequent experiments, we select appropriate parameter reuse ratios for different basic models to ensure that the accuracy loss remains below 5%.

Fig. 5 provides a visualization of AI model parameter reuse, employing models 1-3 in the repository as an example. These models are fine-tuned from the basic Inception-v3 model with parameter reuse ratio of 33.41%, resulting in both reused and individual PBs. If an edge node needs to cache models 1-3, only one copy of the reused PBs is stored. Moreover, we adopt the actual sizes of PBs (ranging from 3.71 KB to 24.31 MB) to derive the numerical results, capturing their distinct storage requirements on the edge nodes.

C. Learning Performance Evaluation

According to Theorem 1, Fig. 6 depicts the upper bound of Q-value approximation error with different τ_0 and ξ . We set $B_{\mathbf{s}} = 1$, $B_{\mathbf{d}} = 1$, $B_r = 1$ due to state/action/reward normalization, $\psi_{\text{in}}^{\max} = 0.5$, $\psi_{\text{re}}^{\max} = 0.5$, $\psi_{\text{out}}^{\max} = 0.5$ to satisfy the echo state property, and $\varsigma = 0.1$, $\phi_{\text{in}}^{\max} = 0.5$, $\phi_{\text{out}}^{\max} = 10$, $U_{\varsigma}^{\text{DRQN}} = 201$, $L_{\varsigma}^{\text{DRQN}} = 46.2$ based on [37]. To minimize the Q-value approximation error, we select the optimal hyperparameters $\tau_0 = 0.8$ and $\xi = 1.12$ for data augmentation experience replay.

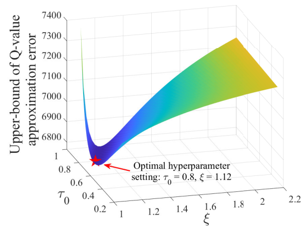
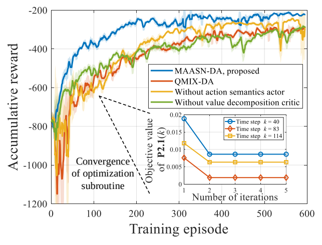
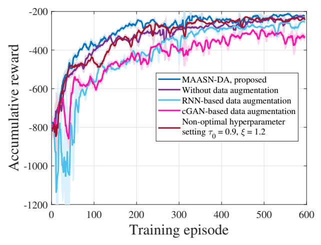


Fig. 6. Upper-bound of Q-value approximation error versus τ_0 and ξ.



(a) Comparison of different training algorithms.



(b) Comparison of different data augmentation strategies.

Fig. 7. Accumulative reward versus training episode.

Fig. 7 assesses the learning performance of our proposed MAASN-DA through an ablation study. The average values (represented by the curves) and standard deviations (illustrated by the shaded regions) of the cumulative reward are used to evaluate convergence speed and stability, respectively. Baseline learning algorithms are outlined below.

1) QMIX-DA [20], [32]: The actor-critic structure of MAASN-DA is replaced by QMIX, which employs a DRQN to estimate each agent's Q-value for discrete actions, and combines these via a mixing network to generate the global Q-value. For a fair comparison, we also introduce the data augmentation experience replay into QMIX.

2) Without Action Semantics Actor: The actor network for each agent is replaced by a traditional fully-connected neural network with two hidden layers, each containing 256 neurons.

3) Without Value Decomposition Critic: Each agent uses an independent critic network to estimate its individual Q-value without further aggregation or value decomposition.

4) Without Data Augmentation: We remove the ESN for generating synthetic samples, and the neural networks are trained solely using data from environment interactions.

5) RNN-Based Data Augmentation: The ESN is replaced by a RNN with an identical architecture, where all weight parameters are tunable.

6) Conditional Generative Adversarial Network (cGAN)-Based Data Augmentation: We train a cGAN that leverages $\mathbf{v}(k)$ as conditional information to generate $(\bar{r}(k), \bar{\mathbf{s}}(k+1))$, while a discriminator is employed to distinguish real experience data from generated samples.

7) Non-Optimal Hyperparameter Setting: We set $\tau_0 = 0.9$ and $\xi = 1.2$ which are different from their optimal setting to validate the theoretical convergence bound analysis.

In Fig. 7 (a), we observe that MAASN-DA achieves the highest reward value with the fastest convergence speed. Compared to *QMIX-DA*, the proposed approach utilizes an actor-critic structure and invokes Gumbel-Softmax reparameteriza-

TABLE III
Evaluation of Algorithm Running Time Per Step

Parameter settings	**Optimization stages**	**Methods**	
		MAASN-DA	Full CoMP
$N = 6$, $M = 20$	MADRL	0.01562 s	/
	Subroutine	0.6501 s	3.4136 s
$N = 6$, $M = 60$	MADRL	0.01606 s	/
	Subroutine	3.0884 s	60.321 s
$N = 12$, $M = 60$	MADRL	0.02013 s	/
	Subroutine	8.3760 s	366.083 s

tion to ensure the discrete constraint, thereby enhancing the model's ability to learn high-dimensional actions. MAASN-DA outperforms *without action semantics actor* owing to its explicit characterization of actions' mutual influence, which enables edge nodes to discern the diverse impact of different action dimensions on other agents. For *without value decomposition critic*, it is difficult for each agent to evaluate the contribution of its own observation and action on the overall system, leading to lower performance than MAASN-DA. Additionally, the lower right corner of Fig. 7 (a) showcases the convergence of the beamforming optimization subroutine. It is evident that the subroutine converges within a few iterations (on average 2-3) for all time steps, demonstrating its efficiency for embedding into MAASN-DA.

Fig. 7 (b) illustrates that MAASN-DA outperforms baseline data augmentation strategies, verifying the effectiveness of incorporating ESN to boost sample efficiency. We can see that the convergence performance of *RNN-based data augmentation* is even worse than *without data augmentation*. This is because tuning all parameters of the RNN makes it difficult to find optimal weights, thereby slowing down overall convergence. In contrast, ESN focuses exclusively on tuning the output weights, and the weights can be efficiently updated via ridge regression. *cGAN-based data augmentation* fails to achieve satisfactory performance, as it struggles to capture the time-dependent nature of state transitions, thereby exhibiting limited compatibility in producing effective synthetic samples for MADRL. Moreover, when using different τ_0 and ξ in *non-optimal hyperparameter setting*, the degradation in convergence performance aligns with the numerical result in Fig. 6. To conclude, the ablation study corroborates the indispensability of all proposed components in MAASN-DA.

Table III reports the running time of different optimization stages (i.e., MADRL for caching and migration design, and subroutine for beamforming optimization) under varying N and M, where *full CoMP* refers to the traditional setting in which all edge nodes participate in the delivery of each PB. It can be observed that the well-trained MADRL agents incur negligible delays for making a decision, while the running time of the proposed optimization subroutine remains moderate. In particular, when $N = 12$ and $M = 60$, our approach reduces the time complexity by nearly two orders of magnitude compared with *full CoMP*. The reason is that MAASN-DA strategically designs PB migration to manage backhaul latency, ensuring that the number of participating edge nodes in broadcasting each PB does not scale monotonically with N, hence the dimensionality of the beamforming variables remains moderate.

D. System Performance Evaluation

In this subsection, we evaluate the performance of the FGAMCD system. In addition to the previously discussed QMIX-DA, the following benchmark methods are employed for comparison.

1) TrimCaching [27]: This method optimizes AI model caching at edge nodes by exploiting parameter shareability to maximize the cache hit ratio, i.e., the satisfaction of user requests. The caching decisions are optimized using a greedy algorithm. As the original method does not account for active PB migration and CoMP beamforming, the corresponding variables are optimized through our proposed approach.

2) Without Edge Cooperation [28]: In this method, each edge node independently optimizes its model caching based on the requests of its associated users. During the downloading phase, the edge node broadcasts its cached PBs to the associated users, while the PB migration and CoMP are omitted.

3) Time Division Multiple Access (TDMA)-Based Unicasting: The PB caching and migration are optimized by our approach. Then, edge nodes unicast the requested PBs to each user via TDMA. Since the PB delivery for each user is orthogonal, the CoMP beamforming is determined using maximum ratio transmission.

4) Coarse-Grained Caching [10], [11]: Similar to conventional content caching, edge nodes directly select the entire AI models for caching, disregarding the parameter reusability among different models. Afterwards, whole AI models are delivered to users during the downloading phase.

Fig. 8 shows the relationship between model downloading delay and edge storage capacity C_n. As C_n increases, all curves exhibit a downward trend. The rationale is that larger storage capacity allows edge nodes to cache more PBs or models, which facilitates the execution of CoMP broadcasting and improves the downloading channel conditions. Furthermore, the proposed MAASN-DA reduces model downloading delay by 12.20%, 12.24%, 23.30%, 29.74%, and 67.86%, respectively, compared to the five benchmark methods. *TrimCaching* focuses on maximizing the cache hit ratio but overlooks the interplay between edge model placement and delivery, the resultant caching decisions do not effectively reduce model downloading delay. For *without edge cooperation*, each edge node only serves its associated users and PB migration is not considered, thus missing the performance improvement introduced by CoMP broadcasting. Both *TDMA-based unicasting* and *coarse-grained caching* exhibit high model downloading delays since neither method leverages parameter reusability during the caching and downloading phases, leading to inefficient use of storage and communication resources. Notably, *coarse-grained caching* fails to meet users' QoS requirements when edge storage capacity is limited.

In Fig. 9, one can observe that the model downloading delay monotonically increases as the number of users grows. This is because a larger variety of model requests necessitates edge nodes to store more diverse PBs so as to meet QoS requirements, which limits the performance improvement of CoMP. Nonetheless, the reduction in model downloading delay achieved by MAASN-DA remains significant, with improvements of 13.06%, 12.44%, 25.38%, 42.85%, and 82.96% over

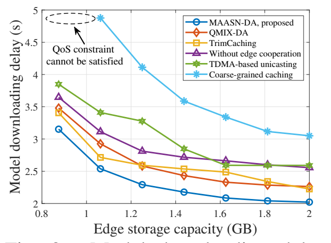


Fig. 8. Model downloading delay versus edge storage capacity.

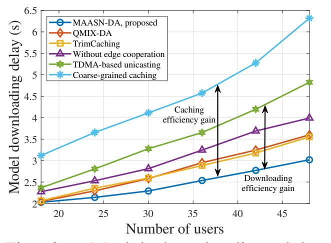


Fig. 9. Model downloading delay versus number of users.

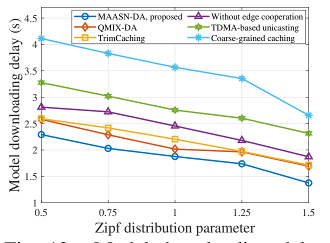


Fig. 12. Model downloading delay versus Zipf distribution parameter.

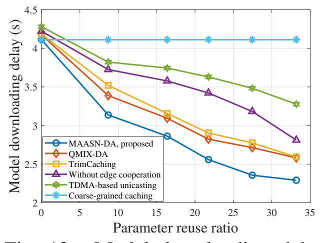


Fig. 13. Model downloading delay versus parameter reuse ratio.

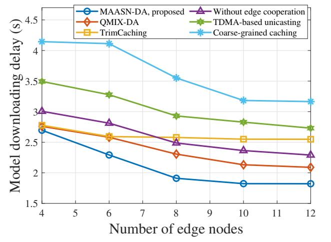


Fig. 10. Model downloading delay versus number of edge nodes.

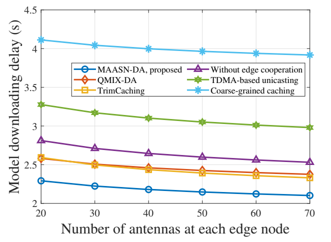


Fig. 11. Model downloading delay versus number of antennas.

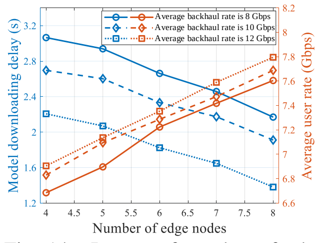


Fig. 14. Impact of number of edge nodes and average backhaul rate.

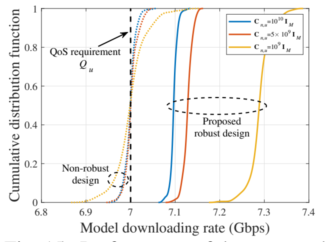


Fig. 15. Performance of the proposed robust CoMP beamforming design.

the five benchmark methods, respectively. More insightful, it is found that the proposed FGAMCD system achieves a two-fold performance gain compared to traditional systems, owing to the exploitation of parameter reusability. The *caching efficiency gain* is realized by eliminating redundant caching of reusable PBs at edge nodes, while the *downloading efficiency gain* is achieved by broadcasting overlapping PBs to simultaneously fulfill the requests of multiple users. In addition, fully unlocking this two-fold performance gain requires dedicated PB caching, migration, and beamforming design, which is effectively implemented by the proposed MAASN-DA.

Figs. 10-11 evaluate the model downloading delay under different network scales. On the one hand, increasing the number of edge nodes N enhances the caching diversity and CoMP gains, while the reduction in downloading delay gradually diminishes when N becomes sufficiently large owing to the limitation of backhaul rates. On the other hand, equipping edge nodes with more antennas provides higher spatial multiplexing gains, thereby enhancing downlink rates for PB broadcasting. Furthermore, our MAASN-DA consistently outperforms the baseline methods under large-scale network settings.

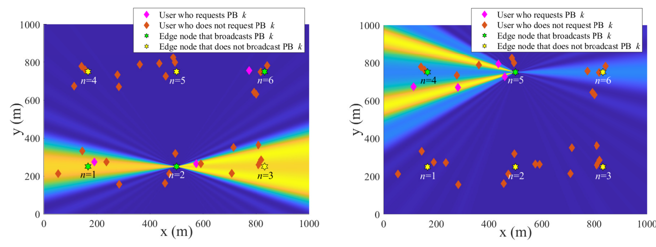
Fig. 12 presents a sensitivity analysis with different Zipf distribution parameters ι, showing that the model downloading delay decreases as ι increases. The rationale is that a larger ι concentrates user requests on hotspot AI models, enabling the PBs constituting these models to be cooperatively cached across multiple edge nodes, thereby facilitating CoMP. This result confirms the robustness of MAASN-DA in adapting to diverse user request patterns.

Fig. 13 illustrates the delay performance of different schemes under varying parameter reuse ratios across AI models. Notably, the proposed FGAMCD framework inherently accommodates different reuse ratios by adjusting the input information $\{\mathcal{K}, \mathcal{K}_j : \forall j\}$, without requiring any modification to the decision-making process. As expected, low parameter reuse leads to higher model downloading delay due to reduced caching and broadcasting gains, while MAASN-DA degenerates to *coarse-grained caching* when the reuse ratio is zero. Moreover, our approach performs well even under small reuse ratios. For instance, when the reuse ratio is 8.70%, the downloading delay is reduced by 8.00%, 12.19%, 18.75%, 21.83%, and 31.09% compared with the five benchmark schemes.

Fig. 14 delineates the impact of number of edge nodes and average backhaul rate on the system performance. As we observe, an increase in both the number of edge nodes and the backhaul rate enhances PB migration and CoMP broadcasting, thereby improving the average user rate and reducing model downloading delay. This result further validates the scalability of the proposed MAASN-DA approach, corroborating its ability to learn the cooperative caching and migration policies, while adapting to varying environmental factors such as the number of edge nodes and the backhaul link conditions.

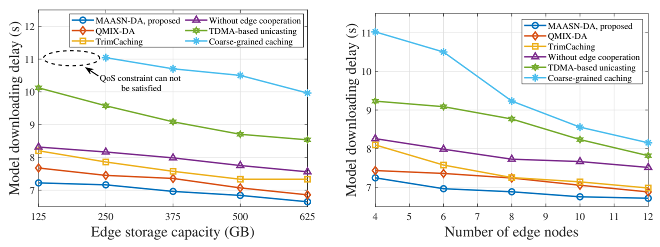
In Fig. 15, we plot the cumulative distribution function (CDF) of model downloading rates across 200 channel error realizations. To emphasize the advantages of the proposed robust beamforming approach, we compare it with a non-robust design, in which the QoS constraint in (9b) is simplified as $\tilde{R}_u(k) \geq \mathbb{I}\{k \in \mathcal{K}_{r_u}\} Q_u,\ \forall k, u$, with $\tilde{R}_u(k)$ being computed based on the estimated CSI. As shown in this figure, the downloading rates achieved by our robust design consistently exceed Q_u under different channel error settings $\mathbf{C}_{n,u}$. In contrast, the non-robust design may violate the QoS constraints due to the fluctuations in channel errors. This highlights the critical importance of robust beamforming in ensuring QoS requirements during AI model downloading, particularly in practical edge networks with CSI uncertainty.

Fig. 16 portrays the beampatterns obtained by the proposed robust CoMP beamforming subroutine, where the beampattern of each edge node n is calculated by $\left|\mathbf{z}^{\mathrm{H}}(\boldsymbol{\theta})\mathbf{w}_n(k)\right|^2$ with $\mathbf{z}(\boldsymbol{\theta})$ being the steering vector towards direction $\boldsymbol{\theta}$. When broadcasting PB $k = 40$, we find that edge node $n = 2$ not only serves its associated user but also radiates power towards the user associated with a neighboring edge node. This is owing to the edge nodes share the PB requested by



(a) $n = 2$, $k = 40$. (b) $n = 5$, $k = 370$.

Fig. 16. Beampatterns of edge nodes for broadcasting different PBs.



(a) Varying edge storage capacity. (b) Varying number of edge nodes.

Fig. 18. Model downloading delay evaluation using fine-tuned Llama2-7B and Llama2-13B models.

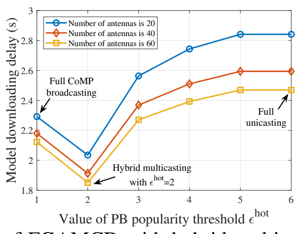


Fig. 17. Integration of FGAMCD with hybrid multicasting.

users, hence CoMP is executed to boost the downloading rate. Analogously, two edge nodes participate in the broadcasting of PB $k = 370$, and the beampattern is appropriately steered towards the requesting users.

In Fig. 17, we examine the integration of FGAMCD with hybrid multicasting. To be specific, we introduce a PB popularity threshold ϵ^{hot}. When the number of users requesting a PB exceeds ϵ^{hot}, CoMP broadcasting across multiple edge nodes is employed; otherwise, the PB is delivered via unicasting from the associated or nearby edge node. As can be observed, for different edge antennas settings, hybrid multicasting with $\epsilon^{\text{hot}} = 2$ achieves lower model downloading delay compared to full CoMP broadcasting. The reason is that the transmission mode is adaptively selected based on the popularity of each PB, thereby enhancing the flexibility of FGAMCD and highlighting a promising research direction for future exploration.

E. Extension to LLM Caching and Downloading

In this subsection, we extend the evaluation of MAASN-DA to billion-parameter LLMs. Specifically, we fine-tune $J = 20$ LLMs based on Llama2-7B and Llama2-13B using the LLaMA-Factory platform [44]. Starting from the pre-trained models on HuggingFace, we freeze the embedding layers and a certain number of initial decoder layers, while fine-tuning the remaining decoders on different subsets of the LIMA [45] and Dolly 15k [46] datasets. The perplexity (PPL) evaluation [44] provided by LLaMA-Factory is employed to measure the performance of the fine-tuned models. Experiments indicate that freezing 28 decoder layers (corresponding to reusable PBs) in Llama2-7B and 35 decoder layers in Llama2-13B leads to a PPL increase of less than 5, which is considered acceptable. To accommodate the incremental storage requirements of LLMs, we adjust the default parameter setting to $K = 285, C_n = 375$ GB, $B = 4 \times 10^4$ MHz, $R^{\text{bac}}_{n,m}(k) \in [3.2, 4.8] \times 10^3$ Gbps, thereby ensuring the feasibility of the solution.

Under the LLM setup, Fig. 18 (a)-(b) illustrate the model downloading delays with respect to edge storage capacity and the number of edge nodes, respectively, which follow consistent trends with those observed in Figs. 8 and 10. Numerical results demonstrate that the proposed MAASN-DA outperforms *QMIX-DA*, *TrimCaching*, *without edge cooperation*, *TDMA-based unicasting*, and *coarse-grained caching* by 5.68%, 8.80%, 14.7%, 30.52%, and 50.86%, respectively, when $N = 6$ and $C_n = 375$ GB. Owing to the pronounced parameter reusability of LLMs, these results corroborate the applicability of MAASN-DA in improving the caching and downloading efficiency for edge LLM deployment.

VI. Conclusion

In this work, we proposed the FGAMCD system to achieve adaptive and low-latency model downloading for on-device AI inference. To unleash the full potential of this system, PB caching, migration, and broadcasting beamforming were jointly optimized to minimize the total model downloading delay, satisfying user QoS requirements, edge storage capacity and transmission power constraints. We developed MAASN-DA, an improved MADRL framework, to enable this joint optimization in a distributed manner. Specifically, MAASN-DA facilitated cooperative decision-making for PB caching and migration via action semantics actor as well as value decomposition critic networks. Moreover, it leveraged an ESN to generate synthetic training samples, thereby enriching the replay buffer and accelerating policy training. We also designed a robust CoMP beamforming subroutine to optimize broadcasting beamformers under CSI uncertainty, which was integrated into the reward calculation to complete the MAASN-DA training loop. Our theoretical analysis showed the convergence of both actor and critic networks, while providing guidelines for configuring learning hyperparameter. In our simulations, the ablation study confirmed the superior learning performance of the MAASN-DA compared to existing MADRL baselines. Furthermore, the FGAMCD system achieved a two-fold performance gain over conventional coarse-grained caching and TDMA-based unicasting schemes, effectively reducing model downloading delay by 29.74% to 67.86%.

Appendix A
Proof of Theorem 1

According to Theorem 4.5 in [41], when the training process only incorporates real samples, the approximation error

$$\begin{aligned}\left|\bar{Q}^{*}-Q^{*}\right| &\overset{(a)}{=} \left|\bar{r}(k)-r(k)+\gamma\left[Q\left(\bar{\mathbf{s}}(k+1),\bar{\mathbf{d}}(k+1)\right)-Q\left(\mathbf{s}(k+1),\mathbf{d}(k+1)\right)\right]\right| \\ &\le \left|\bar{r}(k)-r(k)\right|+\gamma\left|Q\left(\bar{\mathbf{s}}(k+1),\mathbf{d}(k+1)\right)-Q\left(\mathbf{s}(k+1),\mathbf{d}(k+1)\right)\right| \\ &= \left|\bar{r}(k)-r(k)\right|+\gamma\left|\boldsymbol{\omega}_{\text{out}}\left[\tanh\left(\boldsymbol{\omega}_{\text{in}}\bar{\mathbf{v}}(k+1)+\boldsymbol{\omega}_{\text{re}}\mathbf{p}(k)\right)-\tanh\left(\boldsymbol{\omega}_{\text{in}}\mathbf{v}(k+1)+\boldsymbol{\omega}_{\text{re}}\mathbf{p}(k)\right)\right]\right| \\ &\le \left|\bar{r}(k)-r(k)\right|+\gamma\left|\boldsymbol{\omega}_{\text{out}}\left[\boldsymbol{\omega}_{\text{in}}\left(\bar{\mathbf{s}}(k+1)-\mathbf{s}(k+1)\right)\right]\right| \\ &\le \left|\bar{r}(k)-r(k)\right|+\gamma\phi_{\text{out}}^{\max}\phi_{\text{in}}^{\max}\left\|\bar{\mathbf{s}}(k+1)-\mathbf{s}(k+1)\right\| \overset{(b)}{\le} \xi\left(1+\gamma\phi_{\text{out}}^{\max}\phi_{\text{in}}^{\max}\right) \end{aligned} \tag{40}$$

between Q_{π_E} and Q^* is upper-bounded by

$$\mathbb{E}_{\Xi}\left[\left|Q_{\pi_E}-Q^{*}\right|\right] \le \frac{2\vartheta_{\Xi,\Omega}\gamma}{(1-\gamma)^2}\sqrt{\nu^{\max}}+\frac{4\gamma^{E+1}}{(1-\gamma)^2}B_r, \tag{38}$$

where Ξ signifies a fixed distribution of the Dec-POMDP in Section III-B, $\vartheta_{\Xi,\Omega}$ denotes the concentration coefficient determined by Ξ and Ω. $\nu^{\max}=\max_{e\in\mathcal{E}}\mathbb{E}_{\Omega}\left[\left|\tilde{Q}_e-T\tilde{Q}_{e-1}\right|\right]$ represents the maximum one-step error, $\tilde{Q}_e$ is the Q-value in the e-th episode, T is the Bellman optimality operator. Since our work adopts the ESN to generate synthetic samples, we define the optimal Q-value with respect to both the real and synthetic samples as $\bar{Q}^*$, then we modify (38) into

$$\begin{aligned}\mathbb{E}_{\Xi}\left[\left|Q_{\pi_E}-Q^{*}\right|\right] &= \mathbb{E}_{\Xi}\left[\left|Q_{\pi_E}-\bar{Q}^{*}+\bar{Q}^{*}-Q^{*}\right|\right] \\ &\le \mathbb{E}_{\Xi}\left[\left|Q_{\pi_E}-\bar{Q}^{*}\right|\right]+\mathbb{E}_{\Xi}\left[\left|\bar{Q}^{*}-Q^{*}\right|\right] \\ &\le \frac{2\vartheta_{\Xi,\Omega}\gamma}{(1-\gamma)^2}\nu^{\max}+\frac{4\gamma^{E+1}}{(1-\gamma)^2}B_r+\mathbb{E}_{\Xi}\left[\left|\bar{Q}^{*}-Q^{*}\right|\right].\end{aligned} \tag{39}$$

Thereafter, we further express $\left|\bar{Q}^*-Q^*\right|$ in (40), where (a) follows the Bellman equation. $\bar{\mathbf{d}}(k+1)$ is the action output by the actor under $\bar{\mathbf{s}}(k+1)$, whose Q-value is higher than that of taking $\mathbf{d}(k+1)$ under $\bar{\mathbf{s}}(k+1)$. $\phi_{\text{in}}^{\max}$ and $\phi_{\text{out}}^{\max}$ are the bounds of $\|\boldsymbol{\omega}_{\text{in}}\|$ and $\|\boldsymbol{\omega}_{\text{out}}\|$, respectively. (b) is owing to the data quality selection in (17). By substituting (40) into (39), the error bound is given by

$$\begin{aligned}&\mathbb{E}_{\Xi}\left[\left|Q_{\pi_E}-Q^{*}\right|\right] \\ &\le\vartheta_{\Xi,\Omega}\left[\frac{2\gamma}{(1-\gamma)^2}\sqrt{\nu^{\max}}+\xi\left(1+\gamma\phi_{\text{out}}^{\max}\phi_{\text{in}}^{\max}\right)\right]+\frac{4\gamma^{E+1}}{(1-\gamma)^2}B_r.\end{aligned} \tag{41}$$

To proceed with, we derive an upper-bound of $\nu^{\max}$. Specifically, Theorem 2 in [37] is extended as follows

$$\begin{aligned}\nu^{\max} \le& 4\max\left\{B_r/(1-\gamma)\ -\varsigma L_{\varsigma}^{\text{DRQN}},0\right\}^2 \\ &+D_1V^2\ln U_{\varsigma}^{\text{DRQN}}/K'\ +D_2V^2\varsigma,\end{aligned} \tag{42}$$

where ς is a positive constant, K' denotes the number of training samples in each episode. $D_1=8\sqrt{2K'}+256/V$, $D_2=4\sqrt{2K'}+52$, $V=B_r/(1-\gamma)$. $L_{\varsigma}^{\text{DRQN}}$ and $U_{\varsigma}^{\text{DRQN}}$ are the upper- and lower-bounds of the exterior ς-number of the DRQN, respectively, whose specific expressions can be seen in Appendix B of [37].

Our subsequent aim is to characterize K' by taking into account both real and synthetic samples. For tractable analysis, we only consider the effect of the maximum initial number of synthetic samples $\tau_0 K$ and neglect its decay with training episodes. Therefore, K' is given by

$$\begin{aligned}K' &= K+\tau_0 K\Pr\left\{\left\|\left(\bar{r}(k),\bar{\mathbf{s}}(k+1)\right)-\left(r(k),\mathbf{s}(k+1)\right)\right\|\le\xi\right\} \\ &\overset{(a)}{\ge} K+\tau_0K\left(1-\mathbb{E}\left[\left\|\left(\bar{r}(k),\bar{\mathbf{s}}(k+1)\right)-\left(r(k),\mathbf{s}(k+1)\right)\right\|\right]/\xi\right) \\ &\ge K\Big(1+\tau_0-\frac{\tau_0}{\xi}\{\mathbb{E}\left[\left\|\left(\bar{r}(k),\bar{\mathbf{s}}(k+1)\right)\right\|\right] \\ &\qquad+\mathbb{E}\left[\left\|\left(r(k),\mathbf{s}(k+1)\right)\right\|\right]\}\Big),\end{aligned} \tag{43}$$

where (a) is due to the Markov inequality. Since $(\bar{r}(k),\bar{\mathbf{s}}(k+1))$ is output by the ESN, we can bound it as below

$$\begin{aligned}\|\mathbf{q}(k)\| &= \left\|\tanh\left(\eta_{\text{in}}\mathbf{v}(k)+\eta_{\text{re}}\mathbf{q}(k-1)\right)\right\| \\ &\le \left\|\eta_{\text{in}}\mathbf{v}(k)+\eta_{\text{re}}\mathbf{q}(k-1)\right\| \\ &\le \|\eta_{\text{in}}\|\,\|\mathbf{v}(k)\|+\|\eta_{\text{re}}\|\,\|\mathbf{q}(k-1)\| \\ &\overset{(a)}{\le} \psi_{\text{in}}^{\max}\sqrt{B_{\mathbf{s}}^2+B_{\mathbf{d}}^2}+\psi_{\text{re}}^{\max}\|\mathbf{q}(k-1)\|,\end{aligned} \tag{44}$$

where (a) follows Assumption 1. By calculating the sum of geometric series with $\|\mathbf{q}(0)\|=0$, we have

$$\begin{aligned}\|\mathbf{q}(k)\| &\le \psi_{\text{in}}^{\max}\sqrt{B_{\mathbf{s}}+B_{\mathbf{d}}}\sum_{i=0}^{k-1}\left(\psi_{\text{re}}^{\max}\right)^i \\ &= \psi_{\text{in}}^{\max}\sqrt{B_{\mathbf{s}}^2+B_{\mathbf{d}}^2}\frac{1-\left(\psi_{\text{re}}^{\max}\right)^k}{1-\psi_{\text{re}}^{\max}}.\end{aligned} \tag{45}$$

Thus, the output of the ESN is bounded by

$$\begin{aligned}\left\|\left(\bar{r}(k),\bar{\mathbf{s}}(k+1)\right)\right\| &\le \left\|\eta_{\text{out}}\mathbf{q}(k)\right\| \le \left\|\eta_{\text{out}}\mathbf{q}(K)\right\| \\ &\le \psi_{\text{out}}^{\max}\psi_{\text{in}}^{\max}\sqrt{B_{\mathbf{s}}^2+B_{\mathbf{d}}^2}\frac{1-\left(\psi_{\text{re}}^{\max}\right)^K}{1-\psi_{\text{re}}^{\max}}.\end{aligned} \tag{46}$$

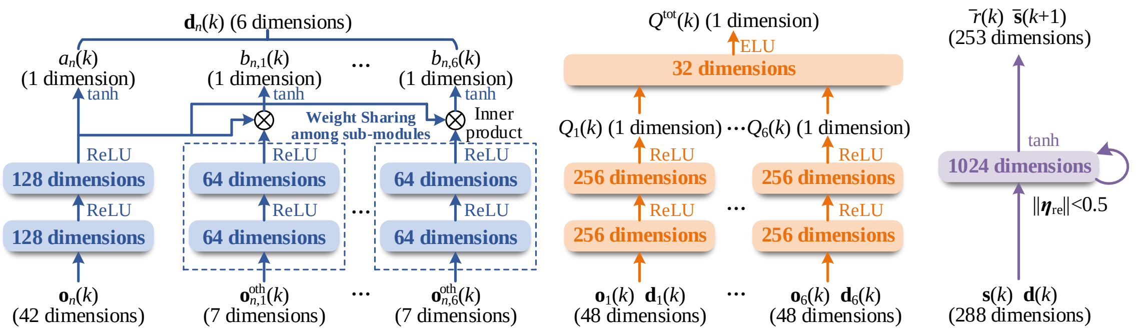
Besides, we can obtain from Assumption 1 that $\left\|\left(r(k),\mathbf{s}(k+1)\right)\right\|\le\sqrt{B_r^2+B_{\mathbf{s}}^2}$, then K' is further derived in (47). Plugging (47) into (42), and (42) into (41), we can derive Theorem 1. This completes the proof.

Appendix B
Proof of Theorem 2

Since the CoMP beamforming subroutine is executed in each time step k of **Algorithm 1**, we first prove that the beamforming problem **P2**(k) can be solved within a finite number of iterations. Denote $f(\mathbf{W})$ and $f^{\text{up}}(\mathbf{W})$ as the objective value in (32) and (33) for a feasible solution $\mathbf{W}$, respectively. Given $\bar{\mathbf{W}}(k)$ from the previous iteration, by solving the **P2.2**(k), we have

$$f\left(\bar{\mathbf{W}}(k)\right)\overset{(a)}{=}f^{\text{up}}\left(\bar{\mathbf{W}}(k)\right)\overset{(b)}{\ge}f^{\text{up}}\left(\mathbf{W}(k)\right)\overset{(c)}{\ge}f\left(\mathbf{W}(k)\right), \tag{48}$$

$$K' \geq K\left(1+\tau_0-\frac{\tau_0}{\xi}\left\{\psi_{\text{out}}^{\max}\psi_{\text{in}}^{\max}\sqrt{B_{\mathbf{s}}^2+B_{\mathbf{d}}^2\frac{1-\left(\psi_{\text{re}}^{\max}\right)^K}{1-\psi_{\text{re}}^{\max}}}+\sqrt{B_r^2+B_{\mathbf{s}}^2}\right\}\right). \tag{47}$$


$\mathbf{d}_n(k)$ (6 dimensions)
$a_n(k)$ (1 dimension)
$b_{n,1}(k)$ (1 dimension)
$b_{n,6}(k)$ (1 dimension)
tanh
Weight Sharing among sub-modules
Inner product
ReLU
128 dimensions
64 dimensions
$\mathbf{o}_n(k)$ (42 dimensions)
$\mathbf{o}^{\text{oth}}_{n,1}(k)$ (7 dimensions)
$\mathbf{o}^{\text{oth}}_{n,6}(k)$ (7 dimensions)
$Q^{\text{tot}}(k)$ (1 dimension)
ELU
32 dimensions
$Q_1(k)$ (1 dimension) ··· $Q_6(k)$ (1 dimension)
256 dimensions
$\mathbf{o}_1(k)$ $\mathbf{d}_1(k)$ (48 dimensions)
$\mathbf{o}_6(k)$ $\mathbf{d}_6(k)$ (48 dimensions)
$\bar{r}(k)$ $\bar{\mathbf{s}}(k+1)$ (253 dimensions)
tanh
1024 dimensions
$\|\boldsymbol{\eta}_{\text{re}}\|<0.5$
$\mathbf{s}(k)$ $\mathbf{d}(k)$ (288 dimensions)

(a) Actor network of agent n. (b) Critic network. (c) Echo state network.

Fig. 19. Neural network architectures of MAASN-DA.

where (a) holds owing to the approximation in (20) is tight with given point $\bar{\mathbf{W}}(k)$, (b) is attributed to that **P2.2**(k) is addressed optimally with solution $\mathbf{W}(k)$, (c) follows that f^{up} is a upper bound of f due to the linearization. (48) indicates that f is non-increasing after each optimization. In addition, according to [40], when **P2.1**(k) with regularization term is solved, the attained $\mathbf{W}(k)$ is a KKT point of **P2**(k). This ensures that the rank-one constraint (23f) is met, yielding a converged beamforming decision.

Subsequently, we show the convergence of the action semantics actor network. Given the actor of each agent $\pi(\mathbf{o}_n(k);\varphi_n)$, the joint policy can be expressed as $\pi = \prod_{n\in\mathcal{N}} \pi(\mathbf{o}_n(k);\varphi_n)$. After performing gradient descent on the actor loss in (22) to generate new policy π_{new}, we have the following inequality:

$$Q_{\pi_{\text{new}}}(\mathbf{s}(k),\mathbf{d}(k)) \geq Q_{\pi}(\mathbf{s}(k),\mathbf{d}(k)), \forall(\mathbf{s}(k),\mathbf{d}(k))\in\mathcal{S}\times\mathcal{D}. \tag{49}$$

Let $E\to\infty$, we get $Q_{\pi_{\text{E}}} > Q_{\pi}$ for any $\pi_E \neq \pi$. According to Theorem 1, $Q_{\pi_{\text{E}}}$ can converge to Q^* with an error bound, then the joint policy π_E that maximize $Q_{\pi_{\text{E}}}$ is local optimal. Therefore, the minimum loss value characterized by (36) is achieved. This completes the proof.

Appendix C
Neural Network Architectures of MAASN-DA

With default parameter setting in Section V, the neural network architectures of MAASN-DA are shown in Fig. 19.

References

[1] Y. Mao, X. Yu, K. Huang *et al.*, "Green edge AI: A contemporary survey," *Proc. IEEE*, vol. 112, no. 7, pp. 880–911, 2024.
[2] ITU-R, "Framework and overall objectives of the future development of IMT for 2030 and beyond," Nov. 2023. [Online]. Available: https://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2030/Pages/default.aspx
[3] Y. Fu, P. Qin, J. Zhang *et al.*, "Joint AI inference and target tracking at network edge: A hybrid offline-online design for UAV-enabled network," *IEEE Trans. Wireless Commun.*, vol. 23, no. 12, pp. 17 959–17 973, 2024.
[4] M. A. Rahman, "A survey on security and privacy of multimodal LLMs - connected healthcare perspective," in *Proc. IEEE Glob. Commun. Conf. Workshops (GC Wkshps)*, 2023, pp. 1807–1812.
[5] S. Mehta, M. H. Sekhavat, Q. Cao *et al.*, "OpenELM: An efficient language model family with open training and inference framework," 2024. [Online]. Available: https://arxiv.org/abs/2404.14619
[6] K. Huang, H. Wu, Z. Liu *et al.*, "In-situ model downloading to realize versatile edge AI in 6G mobile networks," *IEEE Wireless Commun.*, vol. 30, no. 3, pp. 96–102, 2023.
[7] 3GPP, document TS 22.874, "Study on traffic characteristics and performance requirements for AI/ML model transfer," Jun. 2021. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3721
[8] 3GPP, document TS 22.261, "Service requirements for the 5G system," Mar. 2025. [Online]. Available: https://portal.3gpp.org/ChangeRequests.aspx?q=1&versionId=92077&release=195
[9] G. Qu, Q. Chen, W. Wei *et al.*, "Mobile edge intelligence for large language models: A contemporary survey," *IEEE Commun. Surveys Tuts.*, pp. 1–1, 2025.
[10] W. Fan, Z. Chen, Z. Hao *et al.*, "DNN deployment, task offloading, and resource allocation for joint task inference in IIoT," *IEEE Trans Ind. Informat.*, vol. 19, no. 2, pp. 1634–1646, 2023.
[11] K. Zhao, Z. Zhou, X. Chen *et al.*, "EdgeAdaptor: Online configuration adaption, model selection and resource provisioning for edge DNN inference serving at scale," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5870–5886, 2023.
[12] P. Qin, M. Fu, Y. Fu *et al.*, "Collaborative edge computing and program caching with routing plan in C-NOMA-enabled space-air-ground network," *IEEE Trans. Wireless Commun.*, vol. 23, no. 12, pp. 18 302–18 315, 2024.
[13] P. Qin, Y. Fu, J. Zhang *et al.*, "DRL-based resource allocation and trajectory planning for NOMA-enabled multi-UAV collaborative caching 6G network," *IEEE Trans. Veh. Technol.*, vol. 73, no. 6, pp. 8750–8764, 2024.
[14] W. Fan, Q. Meng, G. Wang *et al.*, "Satellite edge intelligence: DRL-based resource management for task inference in LEO-based satellite-ground collaborative networks," *IEEE Trans. Mobile Comput.*, pp. 1–18, 2025.
[15] J. Yan, S. Bi, and Y.-J. A. Zhang, "Optimal model placement and online model splitting for device-edge co-inference," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 8354–8367, 2022.
[16] S. Tuli, G. Casale, and N. R. Jennings, "SplitPlace: AI augmented splitting and placement of large-scale neural networks in mobile edge environments," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5539–5554, 2023.
[17] J. Chen, X. Liang, J. Xue *et al.*, "Evolution of RAN architectures toward 6G: Motivation, development, and enabling technologies," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 3, pp. 1950–1988, 2024.
[18] M. Jafri, S. Srivastava, N. K. D. Venkategowda *et al.*, "Cooperative hybrid transmit beamforming in cell-free mmwave MIMO networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 5, pp. 6023–6038, 2023.

[19] J.-M. Liang, S. Mishra, and I.-C. Chien, “Enhanced cell clustering and multicast scheduling for energy-efficient 5G/B5G MBSFN networks,” *IEEE Internet Things J.*, pp. 1–1, 2025.

[20] J. Chen, K. Zhai, Z. Wang *et al.*, “CoMP and RIS-assisted multicast transmission in a multi-UAV communication system,” *IEEE Trans. Commun.*, vol. 72, no. 6, pp. 3602–3617, 2024.

[21] N. Babu, C. Masouros, C. B. Papadias *et al.*, “Precoding for multi-cell ISAC: From coordinated beamforming to coordinated multipoint and bi-static sensing,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 10, pp. 14 637–14 651, 2024.

[22] G. Xie, Z. Xiong, R. Xie *et al.*, “Mixture of experts-enabled parallel scheduling and processing for vehicular generative AI services,” *IEEE Trans. Cogn. Commun. Netw.*, pp. 1–1, 2025.

[23] Y. Li, Z. Yi, D. Guo *et al.*, “Joint communication and offloading strategy of CoMP UAV-assisted MEC networks,” *IEEE Internet Things J.*, pp. 1–1, 2025.

[24] Z. Lyu, Y. Li, G. Zhu *et al.*, “Rethinking resource management in edge learning: A joint pre-training and fine-tuning design paradigm,” *IEEE Trans. Wireless Commun.*, vol. 24, no. 2, pp. 1584–1601, 2025.

[25] H. Wu, X. Chen, and K. Huang, “Resource management for low-latency cooperative fine-tuning of foundation models at the network edge,” *IEEE Trans. Wireless Commun.*, pp. 1–1, 2025.

[26] E. Hu, Y. Shen, P. Wallis *et al.*, “LoRA: Low-rank adaptation of large language models,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

[27] G. Qu, Z. Lin, F. Liu *et al.*, “Trimcaching: Parameter-sharing AI model caching in wireless edge networks,” in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2024, pp. 36–46.

[28] H. Wu, Q. Zeng, and K. Huang, “Efficient multiuser AI downloading via reusable knowledge broadcasting,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 8, pp. 10 459–10 472, 2024.

[29] R. Wang, J. Liang, C. Feng *et al.*, “Model ownership protection for healthcare consumer electronics in federated edge learning,” *IEEE Trans. Consumer Electron.*, vol. 71, no. 2, pp. 4616–4627, 2025.

[30] H. Geng, Y. Li, S. Wang *et al.*, “Layer redundancy aware DNN model repository planning for fast model download in edge cloud,” *IEEE Trans. Cloud Comput.*, vol. 13, no. 3, pp. 1038–1049, 2025.

[31] X. Gong, X. Liu, A.-A. Lu *et al.*, “Digital twin of channel: Diffusion model for sensing-assisted statistical channel state information generation,” *IEEE Trans. Wireless Commun.*, vol. 24, no. 5, pp. 3805–3821, 2025.

[32] P. Qin, Y. Fu, K. Wu *et al.*, “Packet routing and energy cooperation for RTU satellite-terrestrial multi-hop network in remote cyber-physical power system,” *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 4, pp. 3585–3597, 2024.

[33] P. Qin, Y. Fu, R. Ding *et al.*, “Competition-awareness partial task offloading and UAV deployment for multitier parallel computational internet of vehicles,” *IEEE Syst. J.*, vol. 18, no. 3, pp. 1753–1764, 2024.

[34] W. Fan, P. Chen, X. Chun *et al.*, “MADRL-based model partitioning, aggregation control, and resource allocation for cloud-edge-device collaborative split federated learning,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 6, pp. 5324–5341, 2025.

[35] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, “Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications,” *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, 2020.

[36] W. Wang, T. Yang, Y. Liu *et al.*, “Action semantics network: Considering the effects of actions in multiagent systems,” 2020. [Online]. Available: https://arxiv.org/abs/1907.11461

[37] H.-H. Chang, N. Mohammadi, R. Safavinejad *et al.*, “Dyna-ESN: Efficient deep reinforcement learning for partially observable dynamic spectrum access,” *IEEE Trans. Wireless Commun.*, pp. 1–1, 2024.

[38] T. Rashid, M. Samvelyan, C. S. de Witt *et al.*, “QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning,” 2018. [Online]. Available: https://arxiv.org/abs/1803.11485

[39] Z. Wei, X. Yu, D. W. K. Ng *et al.*, “Resource allocation for simultaneous wireless information and power transfer systems: A tutorial overview,” *Proc. IEEE*, vol. 110, no. 1, pp. 127–149, 2022.

[40] K. Yang, Y. Shi, W. Yu *et al.*, “Energy-efficient processing and robust wireless cooperative transmission for edge inference,” *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9456–9470, 2020.

[41] Z. Yang, Y. Xie, and Z. Wang, “A theoretical analysis of deep Q-learning,” in *Proceedings of the 2nd Conference on Learning for Dynamics and Control (L4DC)*, 2020, pp. 486–489.

[42] Y. Fu, P. Qin, G. Tang *et al.*, “Joint design of sensing, communication, and computation for multi-UAV-enabled over-the-air federated learning,” *IEEE Trans. Veh. Technol.*, pp. 1–17, 2025.

[43] Z. Zhang, Y. Zhao, H. Li *et al.*, “DVFO: Learning-based DVFS for energy-efficient edge-cloud collaborative inference,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9042–9059, 2024.

[44] Y. Zheng, R. Zhang, J. Zhang *et al.*, “LlamaFactory: Unified efficient fine-tuning of 100+ language models,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Bangkok, Thailand: Association for Computational Linguistics, 2024. [Online]. Available: http://arxiv.org/abs/2403.13372

[45] C. Zhou, P. Liu, P. Xu *et al.*, “LIMA: Less is more for alignment,” *NeurIPS*, vol. 36, 2024.

[46] Databricks, Inc., “databricks-dolly-15k: An open instruction-following dataset,” https://huggingface.co/datasets/databricks/databricks-dolly-15k, 2023.