# Dynamic Edge AI Service Management and Adaptation via Off-Policy Meta-Reinforcement Learning and Digital Twin

Yan Chen*, Hao Yu*, Qize Guo*, Shuyuan Zhao‡, Tarik Taleb*†
*Faculty of Information Technology and Electrical Engineering, University of Oulu, Oulu, 90570 Finland
†Ruhr University Bochum, Bochum, 44801 Germany
‡Zhejiang Lab, Hangzhou, 311121 China
{yan.chen, hao.yu, qize.guo}@oulu.fi, zhaosy@zhejianglab.com, tarik.taleb@rub.de

***Abstract*—Edge computing has promoted various applications driven by artificial intelligence (AI). However, upgrading AI models during system operation may change resource and performance features. Then, the service management controller (SMC) faces an unprecedented environmental condition and has limited prior knowledge, resulting in high probabilities of policy mismatches. With the proliferation of AI applications, it is an urgent necessity that SMCs can adapt to different conditions to ensure quality of service (QoS) and resource efficiency. Therefore, this paper studies the problem of dynamic edge AI service adaptation and formulates it as a multi-task scenario adaptation problem. After that, we proposed an approach based on off-policy meta-reinforcement learning and digital twin (DT) technology. The DT system emulates a set of encountered conditions, and a meta-policy is obtained by interacting with these DTs. The executed policy is initialized as the meta-policy once AI models are upgraded. Then, it adapts to new service conditions by drawing salient information from limited transition contexts collected from a newly encountered environmental condition. Simulation results reveal that our approach can optimize QoS and adapt to different service situations.**

***Index Terms*—Artificial intelligence, Adaptive service management, Meta-reinforcement learning, Digital twin**

## I. Introduction

Edge computing is taking hold in various systems, promoting the implementation of applications driven by artificial intelligence (AI) [1], [2]. However, AI models (AiMs) are still black boxes that cannot model and predict their performance precisely, introducing additional complexity to service management in the edge-cloud (EC) continuum. Besides, with the development of AI technologies and data enrichment, service providers (SPs) may upgrade their AiMs if they can benefit in resource utilization, quality of service (QoS), or security. The upgrade of AiMs reshapes the features of resources and QoS. Then, from a service management perspective, the network environment is also changed, as AiSFs are infrastructure components. As a result, the service management controller (SMC) faces an unprecedented environmental condition, i.e., it takes on the same responsibilities (e.g., task and resource allocations) but in a different environmental condition.

Although existing works have contributed to the dynamic task and edge service management in EC systems, most only studied the dynamics of users' states, e.g., mobility and time-varying requirements. Meanwhile, reinforcement learning (RL) and its variations have been widely exploited for dynamic management. In traditional RL-based approaches, the SMC aims to adapt to an unknown system through interactions. Although the system is dynamic, the key features of the network infrastructure generally remain stable, and the system state transition follows a stable distribution, i.e., the state transition distribution and reward function are typically fixed. However, the upgrade of AiMs changes the resource and performance features, reshaping the distribution of state transition and reward functions. Then, there may be a mismatch between the policy obtained in the previous condition and the new AiSF condition, resulting in uninsured QoS [3], [4]. As a result, existing RL-based approaches must retrain the policies, even starting from scratch, introducing substantial retraining costs.

Furthermore, emerging edge intelligence technologies like model pruning [5] can enhance flexible resource utilization by adapting to dynamic requests and resource states. For example, although the accuracy of a pruned AI model is reduced, it requires fewer resources and can reduce processing latency under the same resource conditions [1]. Then, by pruning parameters at different scales, each AiSF can maintain a set of AiMs with the same functionality but diverse resource and performance features. The model with fewer parameters can be activated to provide service once its achieved accuracy satisfies the requirement. Besides, for a given AiM, higher accuracy can be obtained when improving the quality of input data. Therefore, beyond task and resource allocations, the selection of AiMs and the quality of input data impact QoS in EC systems supporting AI-driven applications [2].

With the popularity of AI-driven applications, it is necessary that SMCs can be adaptive in different AiSF conditions to maintain resource efficiency, QoS, and reliability. Therefore, this paper first formulates the problem of joint data processing method selection, AiSF placement, and model selection in dynamic EC systems. Then, we analyze the process of dynamic model upgrades and formulate its policy adaptive requirement as a multi-task scenario adaptation problem. Then, we proposed an approach by leveraging meta-reinforcement

learning (meta-RL), which enables the policy to be adaptive in different AiSF conditions [6], [7]. Meanwhile, digital twins (DT) are created to emulate the AiSF conditions experienced over time. A meta-policy is obtained through interaction with DTs's emulation. Once a new condition is triggered, the policy is initialized as the meta-policy. After that, the policy adapts to the new condition by inferring salient information about it from the collected contexts of the system transition. Simulation results reveal that our approach can adapt to new conditions after SPs upgrade AiMs and minimize service latency. We summarize our main contributions as follows:

- We study adaptive AI service management in EC systems where users have time-varying requests and SPs irregularly upgrade their AiMs. Then, we formulate the problem of joint data compression method selection, service placement, and AiM selection to minimize latency while ensuring accuracy. We further analyze the adaptive requirement after upgrading AiMs and formulate it as a multi-task adaptation problem.
- Then, we propose an approach based on off-policy meta-RL and DT. We use DT to emulate multiple AiSF conditions for assisting in meta-policy training. In the meta policy, a context encoder can infer the salient information of environments, enabling the policy to be adaptive when presented with a new condition and obtain acceptable performance before re-training.

## II. Preliminary

### A. System Model

As shown in Fig. 1, we consider an EC system where smart devices (ED) $\mathcal{U}$ are responsible for diverse operations. Each ED works under the guidance of a dedicated AiSF. Edge servers (ESs) are deployed to support AiSFs, and each ES is associated with an access point (AP) providing communications. Thus, we use $\mathcal{H}$ to represent both APs and ESs. In addition, a cloud server (CS) can tackle complex responsibilities, like model training and releasing edge load.

Each time, an ED $u$ captures necessary data and compresses it by one of the available data pre-processing methods ($\mathcal{M}_u$) for storage and transmission efficiency. Then, the ED processes the task (i.e., compressed data) locally or offloads it to the associated AP. The offloaded task is forwarded to the corresponding AiSF ($\mathcal{F}_u$) that is placed on an ES. Each AiSF maintains multiple AiMs with the same functionality but different QoS and resource features, i.e., $\mathcal{F}_u = \{f_u^1, \cdots\}$. We assume that EDs' requests are stable in a time slot, and the AiSF only activates one AiM to process requests.

An SMC is responsible for optimizing real-time QoS and resource utilization by dynamically managing the placement of AiSFs (TA, i.e., allocating each ED's tasks to an ES), the model selection of AiSFs (MS, i.e., determining which AiM is activated), and the data compress method selection of every ED (DS, e.g., selecting one available data pre-processing method). In addition, SPs update their provided AIMs after collecting enough data or mastering advanced

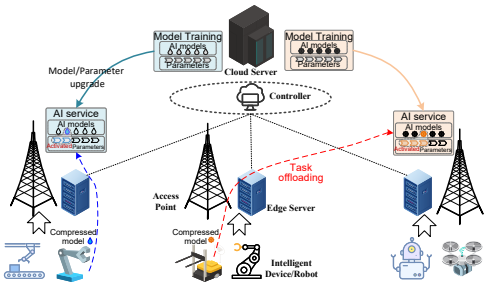


Fig. 1. System Mode

AI technologies. The upgraded AiMs can achieve benefits like reduced resource consumption, improved accuracy, and reduced computation complexity. Once AiMs are upgraded, the SMC needs adaptation to maintain performance under the new QoS and resource features.

### B. QoS Model

When a task is allocated to be processed on ES, its service latency consists of communication and inference delays. The communication delay of task offloading from ED $u$ to connected AP is

$$d_u^c(t) = \frac{V_u^T(t)(\sum_{j=1}^{|\mathcal{M}_u|} x_{u,1}^j(t)\alpha_{u,j}^c) + V_u^R(t)}{\mathcal{B}_u(t)\log_2(1 + p_u(t)g_u(t)/\mathcal{N}(t))}, \quad (1)$$

where $t$ indicates time. $V_u^T(t)$ and $V_u^R(t)$ are the data sizes of $u$'s tasks and results. $x_{u,1}^j(t) \in \{0,1\}$ indicates if selecting the $j$th data processing method of $u$ for data compression, and $\alpha_{u,j}^c$ has a fixed compression ratio. $\mathcal{B}_u(t)$ is the wireless bandwidth allocated to $u$. $|\cdot|$ represents the number of elements in a set. $p_u(t)$ is $u$'s wireless transmission power, and $g_u(t)$ is the corresponding channel gain between $u$ and the connected AP. We employ a 5G path loss model in rural areas [8]. $\mathcal{N}(t)$ is the communication noise. When the task is allocated to a remote ES $h'$ that is not directly associated with the AP $h$ that the ED is connected to, the forwarding latency is

$$d_u^f(t) = \frac{V_u^T(t)(\sum_{j=1}^{|\mathcal{M}_u|} x_{u,1}^j(t)\alpha_{u,j}^c) + V_u^R(t)}{\mathbb{B}_{h,h'}} + \delta_{h,h'}, \quad (2)$$

where $\mathbb{B}_{h,h'}$ and $\delta_{h,h'}$ are separately the transmission rate and propagation delay between AP $h$ and AP $h'$. $\mathbb{B}_{h,h'} = \infty$ and $\delta_{h,h'} = 0$ if $h = h'$.

The inference latency is the time spent on processing the task from receiving a task to generating the corresponding result, which is related to the number of required computations and processing ability of the AiSF, i.e.,

$$d_u^p(t) = \frac{V_u^T(t)\sum_{i=1}^{|\mathcal{M}_u|} x_{u,1}^i(t)\alpha_{u,i}^c \sum_{j=1}^{|\mathcal{F}_u|} x_{u,2}^j(t)\rho_u^j(t)}{\mathcal{P}_u(t)}, \quad (3)$$

where $x_{u,2}^j(t) \in \{0,1\}$ indicates if $u$'s AiSF activates its AiM $f_u^j \in \mathcal{F}_u$ to process tasks. $\rho_u^j(t)$ is the computing intensity of AiM $f_u^j$ regarding the number of computing operations (e.g., FLOP) required for processing per-bit task data. $\mathcal{P}_u(t)$ is the computing ability of $u$'s AiSF (i.e., FLOP per second (FLOPs)) related to corresponding ES ($\mathbb{P}_h^f$), i.e.,

$$\mathcal{P}_u(t) = \sum_{h=1}^{|\mathcal{H}|} x_{u,3}^h(t)\mathbb{P}_h^f, \quad (4)$$

where $x^h_{u,3} \in \{0,1\}$ indicates if $u$'s tasks are processed by ES $h$. We use $x^0_{u,3}=1$ to indicate $u$'s tasks are processed locally, and $x^0_{u,3}=0$ otherwise. When the most lightweight model processes $u$'s tasks locally with speed $\mathcal{P}^f_u$, there is only an inference latency. Thus, the service latency of an ED $u$ is

$$\mathcal{T}_u(t) = \begin{cases} \frac{V^T_u(t)\sum_{i=1}^{|\mathcal{M}_u|} x^i_{u,1}(t)\alpha^c_{u,i}\rho^1_u(t)}{\mathcal{P}^f_u}, & \text{if } x^0_{u,3}=1, \\ d^c_u(t)+d^f_u(t)+d^p_u(t), & \text{otherwise.} \end{cases} \quad (5)$$

In addition to latency, inference accuracy is critical for AI-driven applications. However, modeling and interpreting AiMs are still challenging. The accuracy of an AiM is generally obtained in the training stages. Although the accuracy model of an AiM is unclear, we have common sense that *a better data quality generally requires a larger data size and contributes to higher accuracy, and a more complex model typically makes higher accuracy and computation complexity* [2], [5]. Thus, we express the accuracy of an ED $u$ as

$$A_u(t) = \begin{cases} \sum_{j=1}^{|\mathcal{F}_u|} x^j_{u,2}(t)\mathcal{G}^j_{u,2}(\mathcal{G}_{u,1}(\boldsymbol{x_{u,1}}),t), & \text{if } x^0_{u,3}=0, \\ \mathcal{G}^1_{u,2}(\mathcal{G}_{u,1}(\boldsymbol{x_{u,1}}),t), & \text{otherwise,} \end{cases} \quad (6)$$

where $\mathcal{G}^j_{u,2}(\cdot,t)$ is the mapping function of AiM $f^j_u$ from the input data quality to accuracy. Without considering the noise signal, the input data quality directly depends on the compression scale of the original data, i.e., the selected data pre-processing method ($\boldsymbol{x_{u,1}}$). We use $\mathcal{G}_{u,1}$ to represent the mapping from DS to data quality. $\mathcal{G}_{u,1}$ and $\mathcal{G}^j_{u,2}$ for an AiMs are generally unable to be modeled precisely. The accuracy should satisfy the ED's minimum real-time requirement, i.e.,

$$\textbf{C1:} \quad A_u(t) \geq \tau_u(t), \forall u, \forall t. \quad (7)$$

### C. Resource consumption model

We consider two basic kinds of resources, i.e., computing and memory resources for running AiMs and caching intermediate data on ES. To avoid resource competition, each AiSF occupies a virtual core that the computing unit can provide. The number of AiSFs instanced on each ES and the memory capacity of each ES are both limited, i.e.,

$$\textbf{C2:} \quad \sum\nolimits_{u=1}^{|\mathcal{U}|} x^h_{u,3}(t) \leq \mathbb{P}^n_h, \forall h, \forall t, \quad (8)$$

and

$$\textbf{C3:} \quad \sum\nolimits_{u=1}^{|\mathcal{U}|} x^h_{u,3}(t)\left(\sum\nolimits_{j=1}^{|\mathcal{F}_u|} x^j_{u,2}(t)\mathcal{R}^j_u(t)\right) \leq \mathbb{R}_h, \forall h, \forall t, \quad (9)$$

$\mathcal{R}^j_u(t)$ is the memory required by the $j$th AiM of $u$'s AiSF at time $t$ to load parameters and cache intermediate data. $\mathbb{P}^n_h$ and $\mathbb{R}_h$ are separately the maximum capacities of ES $h$ in computing cores and memory.

## III. Problem Formulation

For AI-driven applications, the primary objective is to obtain accurate results as quickly as possible. Therefore, we set the object as minimizing the service latency while ensuring accuracy by jointly optimizing DS, TA, and MS of EDs, i.e.,

$$\textbf{P1:} \min_{\boldsymbol{x_{u,1}},\boldsymbol{x_{u,2}},\boldsymbol{x_{u,3}}} \frac{1}{T}\sum_{t=1}^{T}\sum_{u\in\mathcal{U}} \mathcal{T}_u(t) \quad (10)$$

$$s.t. \quad \textbf{C1}, \textbf{C2}, \textbf{C3}, \quad (11)$$

$$\textbf{C4:} \sum\nolimits_{j=1}^{|\mathcal{M}_u|} x^j_{u,1}(t)=1, \forall u\in\mathcal{U}, \forall t, \quad (12)$$

$$\textbf{C5:} \sum\nolimits_{j=1}^{|\mathcal{F}_u|} x^j_{u,2}(t)=1, \forall u\in\mathcal{U}, \forall t, \quad (13)$$

$$\textbf{C6:} \sum\nolimits_{j=0}^{|\mathcal{H}|} x^j_{u,3}(t)=1, \forall u\in\mathcal{U}, \forall t, \quad (14)$$

where **C4** indicates only one data pre-processing method can be selected. **C5** constraints that only one AiM is activated at any time slot. **C6** indicates a task can only be processed locally or by the AiSF placed on one ES. Then, the primary object of the problem is to minimize the long-term expected service latency of all EDs. Besides, there is a need for the agent to adapt to different task conditions when SPs update their AiMs.

## IV. Meta-RL-based Approach

### A. Problem analysis

Once SPs upgrade AiMs, there are variations in state transition and QoS calculation since the AiSFs' features regarding resource and QoS are changed. For example, the upgraded AiMs may consume fewer resources. Then, the system can achieve a higher QoS under the same state and employed action(s). Meanwhile, a different action can be employed to achieve better QoS. Although the SMC undertook the same responsibility after each upgrade of AiMs (i.e., DS, TA, and MS of each ED), it could select different optimal actions and achieve different QoS. After each upgrade, the system runs under the same AiMs until the next upgrade. The resource and QoS features after upgrading AiMs are independent of those before. When working under fixed AiMs, the state transition only depends on the properties of the physical network, the deployed AiMs, and the implemented actions. Therefore, the service management process within the duration of fixed AiMs is a Markov decision process (MDP). However, each upgrade of AiMs would change the distribution of state transitions. Thus, we can describe the investigated adaptive AI service management problem as a multi-task system where each task condition is an MDP, i.e., different state transition and reward functions. Besides, the investigated problem has challenges like heterogeneity, dynamic requests, limited prior knowledge, and time-varying state transition and reward functions.

### B. Accuracy-ensured and latency-minimized user selection

The accumulated resources required by AiSFs assigned to an ES according to a random action may exceed its resource constraints, leading some users' tasks to be re-forwarded to CS. However, selecting users from a set of users to re-forward tasks affects the sum of their QoS. After implementing DS, TA, and MS actions, the system's QoS depends only on each ES's operation. Moreover, the DS and MS actions may result in uninsured QoS. As accuracy is critical, we employ

additional operations to ensure accuracy: (1) changing the DS action to the one with the highest data quality if a user's accuracy can not be satisfied. (2) activating its most complex AiM if the accuracy remains unsatisfactory. After that, an ES $h$ selects part users from allocated users $\mathcal{U}^h$ and re-allocate others' tasks to the CS with the object of minimizing the sum of their latency while satisfying its resource constraints, i.e.,

$$\mathbf{P2:} \min_{\boldsymbol{x}'_{\boldsymbol{u},\mathbf{2}}} \sum_{u\in\mathcal{U}^h} x^h_{u,2}(x^{h,\prime}_{u,2}\mathbf{d}^E_u+(1-x^{h,\prime}_{u,2})\mathbf{d}^C_u), s.t., \mathbf{C2,\ C3}, \quad (15)$$

where $x^{h,\prime}_{u,2}\in\{0,1\}$ indicates if ES $h$ selects $u$ to serve. $\mathbf{d}^E_u$ and $\mathbf{d}^C_u$ are the estimated latency when $u$'s tasks are processed on the ES and CS, respectively. As each ES's resource capacities are fixed, **P2** is a multi-dimensional knapsack problem that can be solved by dynamic programming.

### C. Framework of Meta-RL-based approach

We employ DRL to address joint optimization in DS, TA, and MS and design the state, action, and reward as follows:

**State**: The features of AiMs, ESs, and communications maintain long-term stability. Besides, some are not prior information. Therefore, we only consider the frequently changed state of all EDs as the observation of SMC, i.e.,

$$s_t=\left\{(\mathrm{RAN}_u,\mathcal{B}_u,\Upsilon_u,\mathcal{N}_u,V^T_u,V^R_u,\tau_u)|u\in\mathcal{U}\right\}. \quad (16)$$

We ignore the symbol $t$ for simplification. $\mathrm{RAN}_u(t)$ is the RAN to which $u$ connects. $\Upsilon_u(t)$ is the distance between the user and its connected RAN. We assume EDs only move on the ground and transmit with a fixed power, i.e., $p_u\equiv 20$ dbm.

**Action:** For each ED's task request, the action spaces include DS, TA, and MS. Therefore, the system action is the union of all applications' action in every action space, i.e.,

$$a_t=\{a^{\mathrm{DS}}_u,a^{\mathrm{TA}}_u,a^{\mathrm{MS}}_u|u\in\mathcal{U}\}, \quad (17)$$

where

$$\begin{cases} a^{\mathrm{DS}}_u=[p^{\mathrm{DS}}_{u,m}]_{1\times|\mathcal{M}_u|}, m\in\mathcal{M}_u,\\ a^{\mathrm{TA}}=[p^{\mathrm{TA}}_{u,h}]_{1\times(|\mathcal{H}|+1)}, h\in\mathcal{H}\cup\{0\},\\ a^{\mathrm{MS}}_u=[p^{\mathrm{MS}}_{u,f}]_{1\times|\mathcal{F}_u|}, f\in\mathcal{F}_u. \end{cases} \quad (18)$$

$p^{[\cdot]}_{u,i}$ is the probability of selecting the $i$th candidate in the corresponding action space, which is generated by applying softmax and $\sum a^{[\cdot]}_u=1$. Then, by selecting the maximum probability, we can map $a_t$ to $\{\boldsymbol{x_{u,1}},\boldsymbol{x_{u,2}},\boldsymbol{x_{u,3}}\}$. $p^{TA}_{u,0}$ is the probability that ED $u$ processes its tasks locally.

**Reward**: Following the objective defined in (10), we set the reward after implementing action $a_t$ under state observation $s_t$ as the average service latency of all users, i.e.,

$$r_t=\mathbb{E}_{u\sim\mathcal{U}}\left[\mathcal{T}_u(t)\right], \quad (19)$$

where $\mathbb{E}[\cdot]$ represents expectation.

To deal with the requirement of adaptive optimization in different AiSF conditions, we propose a meta-RL-based approach with the assistance of DT. As shown in Fig. 2, the proposed framework includes two main parts, i.e., the physical system and the DT system [9]. SMC operates a physical policy to manage the physical system. At the beginning of each AiSF condition, the physical policy is initialized as the meta policy and runs to collect interaction data. After that, the physical policy infers the salient information utilizing these collected data and is adapted to new conditions. Finally, SMC executes the adapted physical policy for subsequent management. Once service providers upgrade their AiMs, a DT of the new AiSF conditions is created in the DT system and gradually synchronized with the physical system. Meta-policy is trained through interaction with DTs' emulations.

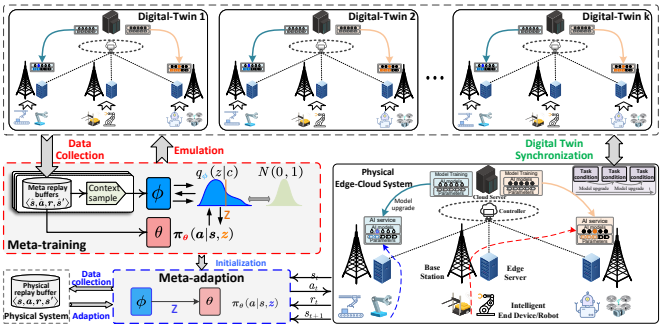


Fig. 2. The framework of meta-RL-enabled dynamic AI service adaptation.

### D. PEARL-based approach

As shown in Fig. 2, we employ an off-policy meta-RL method, i.e., PEARL [7], in which we exploit the SAC in discrete action conditions [10] to generate action and update policy. A context encoder with parameter $\phi$ is introduced, i.e., $q_\phi$. The input of $q_\phi$ is a set of transition contexts collected in an environment condition $k$ so far, i.e., $\boldsymbol{c}^k_{1:N}=(\boldsymbol{s},\boldsymbol{a},\boldsymbol{r},\boldsymbol{s}')$. With these contexts, the encoder generates a latent context $\boldsymbol{z}\sim\boldsymbol{Z}$ that encodes the estimated salient information about the environment (e.g., reward and state transition functions). Then, $\boldsymbol{z}$ is treated as an additional system state. The policy $\pi$ with parameters $\theta$ generates an action $a_t$ for an observed $s_t$ with the assistance of $\boldsymbol{z}$, i.e., $\pi_\theta:(s_t,\boldsymbol{z})\to a_t$, and $q_\phi:\boldsymbol{c}^k\to\boldsymbol{z}$. Thus, the meta-policy can acquire both observed state and inferred salient information about the environment to adapt to different conditions. We can utilize variational inference to estimate the latent distribution $\boldsymbol{Z}$. Then, PEARL is aimed at optimizing the variation lower bound [11], i.e.,

$$\mathbb{E}_{\mathcal{S}}\left[\mathbb{E}_{z\sim q_\phi(\boldsymbol{z}|\boldsymbol{c})}\left[R(\mathcal{S},\boldsymbol{z})+\beta D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{c})||\mathbf{p}(\boldsymbol{z}))\right]\right], \quad (20)$$

where $\mathbf{p}(\boldsymbol{z})$ is a Gaussian prior distribution over $\boldsymbol{Z}$. $R(\mathcal{S},\boldsymbol{z})$ can be different objectives, e.g., minimizing Bellman TD-error. The KL divergence can alternatively be viewed as a variational approximation of the information bottleneck between $\boldsymbol{Z}$ and $\boldsymbol{c}$. This bottleneck limits their mutual information to that essential for adapting to the involved environmental conditions [7].

As DS, TA, and MS are all discrete actions, the probability distribution of each user's action is generated by applying softmax after reshaping and selecting the outputs, i.e., (18). Then, we can calculate the action entropy. A critic with parameter $\psi$ can improve the training stability by estimating the state-action (Q) values, which is updated to minimize TD errors between estimated Q-values, i.e.,

$$\mathcal{L}(\psi)=\mathbb{E}_{\mathcal{D},\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{c})}\left[Q(s_t,a_t,\boldsymbol{z})-\Psi(t)\right], \quad (21)$$

where $\mathcal{D}$ is a batch of sampled experiences. Meanwhile,

$$\Psi(t) = r_t + (1 - \boldsymbol{d})\gamma \mathbb{E}_{s' \sim \{\mathcal{S}, a_t\}} V(s_{t+1}), \tag{22}$$

where $\gamma$ is a discount factor and [1]

$$V(s') := \mathbb{E}_{a' \sim \pi_\theta, \boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{c})} \left[ Q(s', a', \boldsymbol{z}) - \xi \log p_{\pi_\theta}(a'|s') \right] \tag{23}$$

is the modified soft state value of $s_t$. $p_{\pi_\theta}(a_t|s_t)$ is the probability of $\pi_\theta$ selects $a_t$ for $s_t$. $\boldsymbol{d}$ indicates if the system evolves into a terminal state. When policies are fixed, each $Q(s_t, a_t, \boldsymbol{z})$ is a constant. Then, we can rewrite (23) as

$$V(s_{t+1}) = \hat{Q}(s_{t+1}, \hat{a}_{t+1}, \boldsymbol{z}) + \xi \mathbb{H}(\hat{a}_{t+1}) \tag{24}$$

where $\hat{\cdot}$ indicates that the value is generated by the target policy. Target policies are soft-delayed updated from the corresponding policy [12]. $\hat{a}_{t+1}$ is generated by a target actor ($\hat{\pi}_{\hat{\theta}}$) according to $s_{t+1}$. $\mathbb{H}(\hat{a}_{t+1})$ is the entropy of $\hat{a}_{t+1}$. As each action is discrete and independently generated, the entropy of a sub-action $a_t^{[\cdot]} = [p_u^w]_{|\mathcal{U}| \times |\mathcal{W}|}$ in an action $a_t$ is

$$\mathbb{H}(a_t^{[\cdot]}) = -\sum_{u=1}^{|\mathcal{U}|} \sum_{w=1}^{|\mathcal{W}|} p_u^w(t) \log(p_u^w(t)), \tag{25}$$

where $|\mathcal{W}|$ represents candidates' count for each user in this sub-action space. In multi-dimensional conditions, the sum probabilities of all actions equal the number of action dimensions. Thus, we can scale it to the parameter $\xi$.

Similarly, the policy $\pi_\theta$ can be updated by maximizing the estimated Q values according to sampled states, i.e.,

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{s_t \sim \mathcal{D}, \tilde{a}_t \sim \pi_\theta, \\ \boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{c})}} \left[ \xi \log(\pi_\theta(\tilde{a}_t|s_t)) - Q(s_t, \tilde{a}_t, \boldsymbol{z}) \right] \tag{26}$$

$$= \mathbb{E}_{s_t \sim \mathcal{D}, \boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{c})} \left[ -\xi \mathbb{H}(\tilde{a}_t) - Q(s_t, \tilde{a}_t, \boldsymbol{z}) \right], \tag{27}$$

where $\tilde{a}_t$ is regenerated by the updated $\pi_\theta$. The $\xi$ is trainable and is updated during the training by optimizing

$$\mathcal{L}(\xi) = \mathbb{E}_{s_t \sim \mathcal{D}, \boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{c})} [\xi(\mathbb{H}(\tilde{a}_t) + \overline{\mathbb{H}})], \tag{28}$$

where $\overline{\mathbb{H}}$ is a target action entropy set as the negative value of action dimensions. In this work, users in each action space only select one action. Thus, $\overline{\mathbb{H}} = 3 \times |\mathcal{U}|$.

We update the context encoder with an additional KL deviation loss while minimizing the TD-error to update the critic (i.e., (21)), i.e.,

$$\mathcal{L}_{KL} = \beta D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{c}) || \mathbf{p}(\boldsymbol{z})) \tag{29}$$

The procedures for training meta-policy are detailed in Algorithm. 1. The agent adapts to different conditions based on the inference of posterior over $\boldsymbol{Z}$. Thus, the context encoder is updated during meta-training by interacting with different environmental conditions. But it is fixed during adaptation and estimation of $\boldsymbol{Z}$ under new conditions. Besides, we employ a twin-delayed update method to improve stability [12]. We use a negative reward because we aim to minimize latency.

The meta-policy is copied as the physical policy when an upgrade of AiMs is observed. Then, transition contexts are collected and input into the context encoder. Gradually, the generated latent context $\boldsymbol{z}$ can reveal the environment's salient information. After that, the $\pi_\theta$ can generate actions that adapt to the new condition with the assistance of $\boldsymbol{z}$ and obtain expected performance under this condition before updating.

[1]Superscript $[\cdot]'$ equals to subscript $[\cdot]_{t+1}$ in this paper

**Algorithm 1** Meta-policy training based on PEARL

**Input:** Edge-cloud system, meta policy, DT system

**for** $t' = 1, 2, 3, \cdots$ **do**
  **for** $k \in \mathcal{K}$ **do**
    Sample $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{c}^k)$, $a_{k,t} = \pi_\theta(s_{k,t}, \boldsymbol{z})$
    Implement $a_{k,t}$ to get $r_{k,t}$ and observe $s_{k,t+1}$
    Update replay buffer $\mathcal{B}^k$ and $\boldsymbol{c}^k$
  Random select a set of DT environments, i.e., $\mathcal{K}'$
  **for** $t_{\text{epoch}} = 1, 2, \cdots, T_{\text{train}}$ **do**
    **for** $k \in \mathcal{K}'$ **do**
      Sample contexts $\boldsymbol{c}^k$ and $(\boldsymbol{s_{k,t}}, \boldsymbol{s_{k,t+1}})$. $\boldsymbol{z} = q_\phi(\boldsymbol{c}^k)$.
      $\hat{a}_{k,t+1} = \pi_{\hat{\theta}}(s_{k,t+1}, \boldsymbol{z})$, $\tilde{a}_{k,t} = \pi_\theta(s_{k,t}, \boldsymbol{z})$
    $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \hat{\boldsymbol{a}}, \tilde{\boldsymbol{a}}, \boldsymbol{s'}, \boldsymbol{z}) \leftarrow$ union of sampled data
    $\phi$ = Optimizer$(\mathbb{E}_{k \sim \mathcal{K}'} [\mathcal{L}(\psi_i) + \mathcal{L}_{KL}]) |_{i=1,2}$
    $\psi_i$ = Optimizer$(\mathbb{E}_{k \sim \mathcal{K}'} [\mathcal{L}(\psi_i))] |_{i=1,2}$
    **if** $t_{\text{epoch}} \% t_{\lambda_1} == 0$ **then**
      $\xi$ = Optimizer$(\mathbb{E}_{k \sim \mathcal{K}'} [\mathcal{L}(\xi)])$
    **if** $t_{\text{epoch}} \% t_{\lambda_2} == 0$ **then**
      $\boldsymbol{Q} = \min \{Q_{\psi_i}(\boldsymbol{s}, \tilde{\boldsymbol{a}}, \boldsymbol{z})\} |_{i=1,2}$
      $\theta$ =Optimizer$(\mathbb{E}_{k \sim \mathcal{K}'} [\mathcal{L}(\theta)])$
      Soft update $\hat{\theta}, \hat{\psi}_1, \hat{\psi}_2$

## V. Performance Evaluation

We conduct simulations based on Pytorch 2.0 and Gurobi optimizer [13]. We deploy an EC simulation system where features of 4 deployed ESs are set as $\mathbb{P}_h^f$ :{60, 80, 100, 120} GFLOPs, $\mathbb{R}_h$: {24, 24, 32, 32} GB, and $\mathbb{P}_h^n$ : $\{10, 24, 16, 32\}$. We set each AiSF on CS running the most complex AiM and have 400 GFLOPs processing speed. At each time, we randomly generate states of 80 EDs according to uniform distributions of configurations, but the accuracy requirement follows a normal distribution, i.e., $\tau_u \sim N(70, 5^2)$. We randomly generated three groups of AiSFs. Then, the AiSFs of the first group are randomly upgraded twice, i.e., lower complexity, memory consumption, and higher accuracy. AiSFs in the second group are upgraded once. Then, we use these six environments to train meta-policy, and $|\mathcal{K}'| = 2$. After that, we randomly generated another three AiSF conditions to evaluate the updated policies. Table. I details more configurations.

TABLE I
Simulation Configurations

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $\mathbb{B}_{i,j}$ | [8, 20]×10 Mbps | $\mathcal{B}_u$ | [2, 7]×8 MHz |
| $\mathcal{N}_u$ | [-120, -80] dbm | $V_u^T$ | [1000, 1500] KB |
| $\rho_u$ | [300, 3000] FLOP | $\Upsilon_{2d}$ | [30, 100] m |
| $V_u^R$ | [10, 20] KB | $\delta_{i,j}$ | [10, 20] ms |
| $\alpha_u$ | [50, 100]% | $\mathcal{R}_u$ | [256, 2048] MB |
| Optimizer | AdamW | Learning rata | $3e^{-5}$ |
| $\lambda_1, \lambda_2$ | 10, 40 | $T_{\text{train}}$ | 40 |
| batch size | 256 | hidden size | 700 |
| hidden layers | 5 | $\beta$ | 0.1 |

We compare our proposed approach with the traditional DRL method (i.e., SAC). We separately train SAC agents base on only one of the test AiSFs indexed from 1 to 3

(SAC_AiSF) [1], [14]. Besides, we trained the SAC agent based on multiple AiSF conditions [3], [6], including data collected from DTs (SAC_DTs, i.e., same as our approach) and the three test AiSF conditions (SAC_T_M). We employ early-stopping technology to tackle the problem of overfitting.

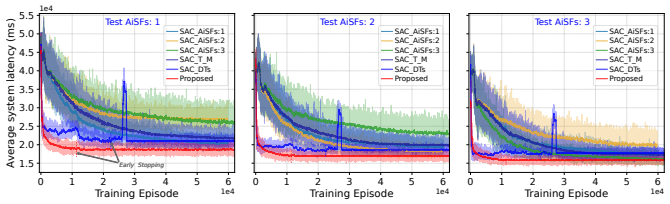


Fig. 3. Average system latency comparison in different testing environments

Fig. 3 displays the average system service latency in three testing AiSF conditions over the training process. We can find that SAC agents trained under a set of specific AiSFs can only converge to suboptimal performance after long-term training. However, they achieve significantly higher latency in other conditions than the agents trained in corresponding conditions, which indicates a policy mismatch occurred. When training in DTs, the agents converge faster since they collect more data and can update parameters more times. Besides, we can find that the proposed approach can adapt to different conditions of AiSFs and minimize service latency before a re-training, although the conditions have never been encountered before. However, the latency achieved by SAC agent training based on DTs is slightly higher than that training in the corresponding conditions because the SAC agent has no adaptation ability. Besides, when training from interactions with DTs, the performance of SAC is unstable as it is easily overfitted to the AiSFs of DTs rather than test AiSFs. Moreover, the convergence of achieved performance is more stable than others. The meta-RL policies can infer latent context information about the environment, improving exploration and making contexts well adapted to these environmental conditions.

## VI. Conclusion

This work studied adaptive edge AI service management in EC systems. We formulated the problem of minimizing service latency while ensuring inference accuracy through joint data processing method selection, task allocation, and AI model selection. Considering the requirement of adaptation under different AiSF features, an approach based on off-policy meta-RL and DT is proposed. The meta-policy runs with a context encoder that enables it to infer salient information about AiSF conditions from collected system transition trajectories. Meanwhile, DTs of different AiSF conditions are leveraged to provide emulations and data for the training, making the meta-policy able to learn some adaptive knowledge from diverse experiences and utilize the acquired skills to adapt to unencountered conditions and enhance exploration. Simulation results reveal that the proposed meta-RL-based approach outperforms the traditional DRL-based methods. It can adapt to new AiSF conditions and obtain acceptable performance prior to further parameter updates.

## Acknowledgment

This work is partially supported by the European Union's Horizon 2020 Research and Innovation Program through the aerOS project under Grant No. 101069732; the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022; the European Union's HE research and innovation program HORIZON-JUSNS-2022 under the 6GSandbox project (Grant No. 101096328); the National Key Research and Development Program of China under Grant No. 2021YFB2900200; and the Research Council of Finland 6G Flagship Programme under Grant No. 346208. This research was also conducted at ICTFICIAL Oy. The paper reflects only the authors' views, and the European Commission bears no responsibility for any utilization of the information contained herein.

## References

[1] C. Deng, X. Fang, and X. Wang, "UAV-Enabled Mobile-Edge Computing for AI Applications: Joint Model Decision, Resource Allocation, and Trajectory Optimization," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5662–5675, Apr. 2023.

[2] Y. Shi, C. Yi, B. Chen, C. Yang, K. Zhu, and J. Cai, "Joint Online Optimization of Data Sampling Rate and Preprocessing Mode for Edge–Cloud Collaboration-Enabled Industrial IoT," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 16 402–16 417, Sep. 2022.

[3] G. Qu, H. Wu, R. Li, and P. Jiao, "Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.

[4] Y. Yuan, G. Zheng, K.-K. Wong, and K. B. Letaief, "Meta-Reinforcement Learning Based Resource Allocation for Dynamic V2X Communications," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8964–8977, Sep. 2021.

[5] B. Fang, X. Zeng, and M. Zhang, "NestDNN: Resource-Aware Multi-Tenant On-Device Deep Learning for Continuous Mobile Vision," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, Oct. 2018, pp. 115–127.

[6] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2021.

[7] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine, "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables," in *Proceedings of the 36th International Conference on Machine Learning (PMLR)*, Long Beach, USA, Jun. 2019, pp. 5331–5340.

[8] ETSI, "Lte;5g; study on channel model for frequency spectrum above 6 ghz (3gpp tr 38.900 version 14.2.0 release 14)," European Telecommunications Standards Institute,3rd Generation Partnership Project (3GPP), Technical Report (TR) 138 900, 06 2017, version 14.2.0.

[9] P. Almasan, M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, D. Perino, D. López, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Network digital twin: Context, enabling technologies, and opportunities," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022.

[10] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019. [Online]. Available: https://arxiv.org/abs/1910.07207

[11] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *International Conference on Learning Representations ICLR'17*, Toulon, France, 2017.

[12] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1587–1596.

[13] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[14] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-Guaranteed Collaborative DNN Inference in Industrial IoT via Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021.