IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 1, 2021, accepted October 31, 2021, date of publication November 2, 2021, date of current version November 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3125008

Repot: Transferable Reinforcement Learning for Quality-Centric Networked Monitoring in Various Environments

YOUNGSEOK LEE[1], WOO KYUNG KIM[2], SUNG HYUN CHOI[2], IKJUN YEOM[2], AND HONGUK WOO[2], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea
[2]Department of Computer Science and Engineering, Sungkyunkwan University, Suwon 16419, South Korea

Corresponding author: Honguk Woo (hwoo@skku.edu)

This work was supported in part by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant 2021-0-00900 and Grant 2021-0-00875, and in part by the ICT Creative Consilience program supervised by the IITP under Grant IITP-2020-0-01821.

ABSTRACT Collecting and monitoring data in low-latency from numerous sensing devices is one of the key foundations in networked cyber-physical applications such as industrial process control, intelligent traffic control, and networked robots. As the delay in data updates can degrade the quality of networked monitoring, it is desirable to continuously maintain the optimal setting on sensing devices in terms of transmission rates and bandwidth allocation, taking into account application requirements as well as time-varying conditions of underlying network environments. In this paper, we adapt deep reinforcement learning (RL) to achieve a bandwidth allocation policy in networked monitoring. We present a transferable RL model Repot in which a policy trained in an easy-to-learn network environment can be readily adjusted in various target network environments. Specifically, we employ *flow embedding* and *action shaping* schemes in Repot that enable the systematic adaptation of a bandwidth allocation policy to the conditions of a target environment. Through experiments with the NS-3 network simulator, we show that Repot achieves stable and high monitoring performance across different network conditions, e.g., outperforming other heuristics and learning-based solutions by 14.5∼20.8% in quality-of-experience (QoE) for a target network environment. We also demonstrate the sample-efficient adaptation in Repot by exploiting only 6.25% of the sample amount required for model training from scratch. We present a case study with the SUMO mobility simulator and verify the benefits of Repot in practical scenarios, showing performance gains over the others, e.g., 6.5% in urban-scale and 12.6% in suburb-scale.

INDEX TERMS Networked monitoring systems, bandwidth allocation, transferable reinforcement learning, domain adaptation, policy transfer, flow embedding, action shaping.

I. INTRODUCTION

In cyber-physical applications, sensing devices operate as data sources of a distributed database in that each continuously sends its status information to a centralized node that evaluates application-specific queries on aggregated information. Such a sensor-based, networked monitoring system requires the status updates to be as timely as possible to maintain the high-quality in query evaluation [1]–[3]. However, due to the nature of existing network infrastructures with inherent restrictions on low-latency data communications, it can be a challenging problem to ensure the timeliness of status updates and information aggregation at all times from numerous sensing devices [4].

In general, the problem has been investigated in several fields of network systems and applications such as the age of information (AoI) [4], [5], decision fusion [6]–[8] sensor networks [9], [10], and Internet of Things (IoT) [11]. Existing studies have normally focused on heuristic strategies on status updates according to given optimization objectives and resource constraints. For example, Jiang *et al.* [5] addressed AoI problems in wireless networks with dynamic channel

The associate editor coordinating the review of this manuscript and approving it for publication was Eyuphan Bulut.

errors by exploiting the closed-form Whittle's index [12] that estimates status tracking accuracy and establishing a heuristic strategy to configure the transmission rates of sensing devices. At the physical network level, Ciuonzo *et al*. [8] presented efficient decision fusion rules over massive multiple-input multiple-output (MIMO) in wireless sensor networks to reduce complexity and improve energy-efficiency by employing a widely-linear statistic [13], [14] and linear filters.

In this paper, we take a learning-based approach for scheduling and optimizing the data transmission and state updates under restrictive network conditions. Deep reinforcement learning (RL) has been recently considered a feasible solution to tackle complex optimization problems in the area of various networked systems, e.g., energy optimization in data centers [15], [16], cluster resource management in cloud computing [17]–[19], video streaming in wireless networks [20], network slicing [21], [22], and others. In the same vein as those RL-based approaches, we address the optimization problem of status updates in networked monitoring by formulating it as successive decisions on bandwidth allocation for multiple sensing devices, which can be modeled in a Markov decision process (MDP) to be learned through RL.

In doing so, we propose a transferable RL model of which the structure is tailored to the characteristics of networked monitoring. Our proposed model is called **Repot** (REinforcement learning POlicy with Transferability). In **Repot**, we train a bandwidth allocation policy in a learnable network environment using conventional RL algorithms (e.g., SAC [23]) and then adapt the policy according to the conditions of a specific target network environment using *flow embedding* and *action shaping* schemes.

The flow embedding scheme is intended to represent each sensor data stream and its relation to the other streams on a common vector space, rendering **Repot** scalable upon a wide variety of observed network states. The action shaping scheme is intended to decompose the action inference on bandwidth allocation into a two-staged procedure with several modules, by which a latent action is generated upon flow embeddings and then its representation can be transformed according to different conditions (e.g., underlying network limitations or spatial characteristics of monitored objects).

These two schemes in **Repot** enable the rapid adaptation of a policy optimized in an easy-to-learn source environment to a target environment, and alleviate the difficulty in achieving an optimal policy across a variety of network scales and environment conditions. For example, collecting 1.6M training samples in a network simulator requires tens or hundreds of days (e.g., in Table 4). That required amount of samples is estimated as a minimum to have a (non-pretrained) model converged in a target environment based on our simulation. In **Repot**, the adaptation schemes can establish a competitive policy sample-efficiently. The learned policy enables high-quality monitoring, showing 14.5∼20.8% higher in quality-of-experience (QoE) than other heuristic and learning-based methods in comparison for a given target environment (in Figure 5). This performance benefit is achieved by the sample efficient adaptation schemes, in which about 100K samples are used to transfer a policy learned in a source environment; that is only 6.25% of what can be originally demanded for model training from scratch (e.g., 1.6M).

As such, **Repot** allows us not only to exploit conventional RL algorithms to robustly establish a policy optimized in a source environment, but also to efficiently adapt the policy to target environments. **Repot** shows robust adaptation performance, comparable to that optimized in a source environment (i.e., within 1% margin in Figure 5).

Furthermore, we present a case study with the SUMO (Simulation of Urban MObility) mobility simulator [24] and demonstrate the applicability of **Repot** in practical network monitoring scenarios. **Repot** achieves performance gains over the other methods, e.g., 6.5% in urban-scale and 12.6% in suburb-scale scenarios (in Figure 10).

In **Repot**, we focus on the modular model structure and policy transferability in RL, which is the first attempt in the context of networked monitoring. The main contributions of this paper are summarized as follows.

- We present a transferable RL model **Repot** by which a bandwidth allocation policy in networked monitoring can be adapted for different network conditions.
- We develop adaptation schemes in **Repot** such as *flow embedding* and *action shaping* that provide scalable state embedding and efficient policy adjustment, respectively.
- We show that **Repot** performs competitively in both source and target environments, compared to other algorithms including a state-of-the-art learning model, through various experiments with NS-3 [25] and a case study of traffic datasets generated by SUMO [24].

The rest of the paper is organized as follows. Section II describes the problem of networked monitoring in different network environments and our RL-based approach to it. Section III presents the modular structure and algorithm of our proposed model with flow embedding and bandwidth allocation modules, and describes the adaptation scheme based on action shaping. Sections IV, V, and VI provide the experiment results, the related research works, and the conclusion, respectively. In addition, Table 1 provides a list

TABLE 1. A list of acronyms.

Acronym	Definition	Acronym	Definition
Repot	REinforcement learning POlicy with Transferability	MEC	Multi-access Edge Computing
NS-3	Network Simulator Ver. 3	SUMO	Sim. of Urban MObility
MIMO	Multiple-input Multiple-output	LTE-U	Long-term Evolution Unlicensed
UAV	Unmanned Aerial Vehicle	AoI	Age of Information
MDP	Markov Decision Process	SAC	Soft Actor-critic
QUAL	Quality Function	ALLOC	Allocation Function
SHAPE	Shaping Function	ADJUST	Adjustment Function
MLP	Multi-layer Perceptron	SSIM	Structural Similarity Index Measure
ARN	Attention-integrated Relevance Network	URLLC	Ultra-reliable Low Latency Communication

TABLE 2. A list of notations.

Symbol	Descriptions	Symbol	Descriptions
	System		
D_i	Sensing device i	N_D	The num. of devices
t	A discrete time-step	I_i^t	The latest updated info. at t by D_i
I^t	Aggregated infor. of all D_i at t	QUAL	A quality func. for monitoring quality eval.
a_i	A bandwidth limit allocated for D_i	$L_{\mathcal{E}}$	The link capacity value given environment $\mathcal{E}$
A	The total BW limit	T	An entire period
	Algorithm		
S^t	A state with all flow states for all D_i at t	$\mathbf{a}^t$	An action (a set of a_i) by a policy at t
r^t	A reward based on QUAL with I^t at t	EMB_ψ	Flow embedding module with parameter ψ
ALLOC_{ϕ_1}	An allocation func. with parameter ϕ_1	ADJUST_{ϕ_2}	An adjustment func. with parameter ϕ_2
SHAPE	A shaping func. with non-trainable parameter	$\mathbf{E}^t$	Flow embeddings by EMB_ψ
$\tilde{\mathbf{a}}^t$	A latent action by ADJUST_{ϕ_1}	Δ^t	Control values by ADJUST_{ϕ_2}
	Implementation		
$\mathcal{D}$	A replay pool for RL	α	A temperature parameter
π_{ϕ_1}	A policy for ALLOC_{ϕ_1}	π_{ϕ_2}	A policy for ADJUST_{ϕ_2}
SAC_m	An SAC structure for $\pi_{\phi_m}, m \in \{1,2\}$	$Q_{\theta_{m,n}}$	A soft Q-func. in SAC_m for $n \in \{1,2\}$
ψ_{Act}	A para. of flow embed. in the actor of SAC_1	ψ_{Crit}	A para. of flow embed. in the critic of SAC_1
$\theta_{m,n}$	A para. of $Q_{\theta_{m,n}}$	$J_\alpha^{(m)}$	An objective func. for α
$J_{\text{Crit}}^{(m,n)}$	A critic objective func. for $\theta_{m,n}, \psi_{\text{Crit}}$ in SAC_m	$J_{\text{Act}}^{(m)}$	An actor objective func. for $\phi_m, \psi_{\text{Act}}$ in SAC_m

of acronyms used in this paper, and Table 2 summarizes notations frequently used in three aspects (system, algorithm, and implementation).

II. OVERALL SYSTEM

In this section, we explain the problem formulation regarding the QoE assurance on networked monitoring in various environments, and describe our approach to the problem.

A. QUALITY-CENTRIC NETWORKED MONITORING

In resource-constrained network environments, we consider a monitoring system in which geographically distributed sensing devices communicate with a server running data-driven applications on aggregated information. The QoE achieved by data-driven applications such as industrial process control, intelligent traffic control, and networked robots is usually dependent on the timeliness of status updates and aggregation. For example, a high-quality map can be created based on real-time data streams from geographically distributed sensing devices [26], [27].

Figure 1 briefly illustrates such a networked monitoring system where a server employs a bandwidth allocation strategy among networked sensor devices to continually ensure high QoE in query evaluation on timely updates from devices. Algorithm 1 represents the overall procedure in a networked monitoring system.

Specifically, each device D_i collects surrounding information about tracked objects and sends it to a server in packets (e.g., multi-access edge computing (MEC) systems [28]). Timely aggregated information from all D_i enables real-time

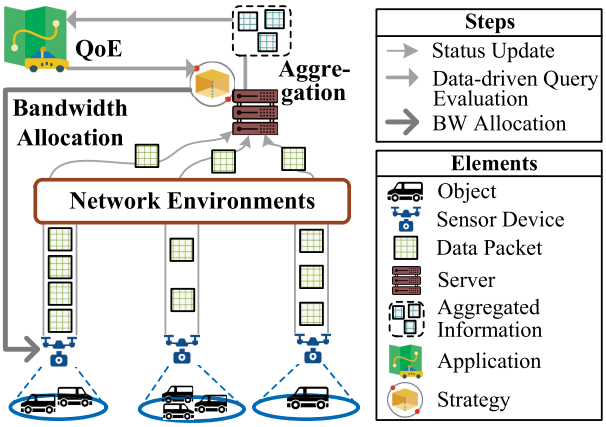


FIGURE 1. Concept of a networked monitoring system: A bandwidth allocation strategy is employed on a server for sensor devices in a networked monitoring system to ensure high QoE of query evaluation over aggregated information. The strategy procedure consists of three steps; (1) status update, (2) data-driven query evaluation, (3) bandwidth allocation.

Algorithm 1 BW Allocation Procedure

1: **while** *Application is working* **do**
2: */* Step 1. Status update */*
3: Sensor devices collect surrounding information
4: Each device sends the collected information (status) to the server
5: */* Step 2. Data-driven query evaluation */*
6: The server evaluates a data-driven query over the aggregated information and achieves QoE
7: */* Step 3. Bandwidth allocation */*
8: A strategy on the server calculates bandwidth limit values of the sensor devices for maximizing QoE
9: The bandwidth limit value is transmitted to each device
10: **end while**

monitoring for $i \in \{1, \ldots, N_D\}$. We represent the latest updated information at time-step t by D_i as I_i^t and the aggregated information of all D_i as

$$I^t = \left(I_1^t, \ldots, I_{N_D}^t\right). \tag{1}$$

In addition, we represent the real-time monitoring quality as a function QUAL that is evaluated on I^t in an application-specific way (e.g., Eq. (21)). Due to resource limitations of underlying network systems, it is non-trivial to always maintain I^t up-to-date and achieve the optimal quality. We assume that bandwidth is a major limited resource affecting the transmission rate of sensor devices. We represent the link capacity as $L_{\mathcal{E}}$ for a network environment $\mathcal{E}$ and the bandwidth allocated for D_i as a_i. Then, we have a resource constraint, $\sum_{i=1}^{N_D} a_i = A \leq L_{\mathcal{E}}$, and formulate an optimization problem such as

$$\underset{a_1^t, \ldots, a_{N_D}^t}{\text{maximize}} \ \text{QoE} = \sum_{t=0}^{T-1} \text{QUAL}(I^t; a_1^t, \ldots, a_{N_D}^t),$$

$$\text{subject to } \sum_{i=1}^{N_D} a_i^t = A \leq L_{\mathcal{E}} \tag{2}$$

where T denotes an entire time-step period, t denotes a discrete time-step, and N_D denotes the number of sensing devices.

Given the formulation, we aim at achieving an RL-based (resource) orchestrator that allocates the bandwidth limit a_i of sensing device D_i to maximize the monitoring quality (QoE) under the limited overall capacity ($L_{\mathcal{E}}$). That is, an RL-based orchestrator takes online network status as an input state and determines $\{a_i^t\}_{i=1}^{N_D}$ at a time-step t, receiving rewards based on achieved QoE.

In this regard, we address the challenging problems of such RL-based orchestrator under a variety of network conditions in terms of scales, configurations, and observation dynamics, which usually affect the performance of the orchestrator deployed in a target environment.

B. RL-BASED ORCHESTRATION

For the optimization problem in Eq. (2), we formulate an RL-based orchestrator in an MDP with a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. An MDP consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a state transition probability $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, a reward $r : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, and a discount factor $\gamma \in [0, 1]$. We assume that $\mathcal{S}$ and $\mathcal{A}$ are continuous and p is unknown.

1) STATE

A state S is represented as

$$S = \{\mathbf{S}_1, \ldots, \mathbf{S}_{N_D}\} \tag{3}$$

where $\mathbf{S}_i = [\mathbf{s}_i^0 = [s_1^0, \ldots, s_d^0]^\intercal, \ldots, \mathbf{s}_i^{-u}] \in \mathbb{R}^{d \times (u+1)}$ denotes the flow states of individual D_i for d distinct features with a history window in $(u + 1)$ time-steps. Note that $\mathbf{s}_i^0$ denotes the current state of device D_i, and $\mathbf{s}_i^{-1}$ denotes its one-step previous state.

In our implementation, we set $d = 3$ in that the features include status update information, whether or not the information varies from the last one, and timestamp, and furthermore, we set the history size $u = 3$.

2) ACTION

An RL-based orchestrator determines an action $\mathbf{a}$ using a ϕ-parameterized policy π_ϕ upon a state S,

$$\mathbf{a} = [a_1, \ldots, a_{N_D}]^\intercal \tag{4}$$

where a_i sets the bandwidth limit of each device D_i.

3) REWARD

A reward r is calculated based on the quality function QUAL(·). For time-step t, we have

$$r = \text{QUAL}(I^t). \tag{5}$$

C. POLICY TRANSFERABLE RL

In **Repot**, the flow embedding scheme is intended to represent the relation of multiple flow states in low dimensional vectors, extracting the historical features from status updates of sensing devices. Given the flow embedding vectors up-to-date as input, actions on bandwidth allocation are calculated through the action shaping scheme.

To achieve the **Repot** model, we employ the two-phase model training; a base model is trained in an easy-to-learn source environment and then its learned policy is adapted to a target environment. Figure 2 illustrates the modular model structure in **Repot**, with the flow embedding and bandwidth allocation modules in the middle and top, respectively. It also represents the two-phase model training, where the left part corresponds to model training in a source environment and the right part corresponds to fine-tuning for adaptation in a target environment. (Phase-1) In a source environment, the flow embedding module and the allocation function in the bandwidth allocation module are trained to establish a general policy on bandwidth allocation upon flow embeddings. (Phase-2) Then, the bandwidth allocation policy optimized in the source environment is fine-tuned to adapt to the network

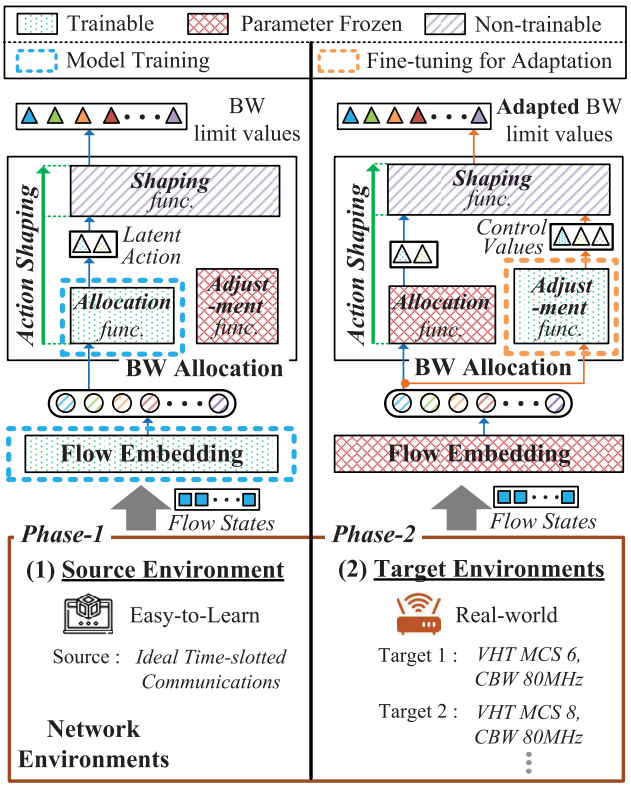


FIGURE 2. Model training and adaptation in source and target environments: The functions in the flow embedding and bandwidth allocation modules are represented in different box patterns to clarify the difference of (Phase-1) model training for a general bandwidth allocation strategy in a source environment and (Phase-2) fine-tuning for adaptation in a target environment. The dotted line boxes in blue represent two functions trained in Phase 1, and the dotted line box in orange represents a function fine-tuned in Phase 2.

Algorithm 2 (Phase-1) Model Training

1: */* Phase-1: Model training in a source environment */*
2: **while** *The training loss is not converged* **do**
3: The flow embedding module computes flow embeddings by taking flow states from devices
4: The allocation function determines a latent action
5: The shaping function transforms the latent action into an action
6: The training loss is calculated from objective functions
7: The parameters of the flow embedding module and allocation function are updated based on the loss to establish a general policy
8: **end while**

Algorithm 3 (Phase-2) Fine-Tuning

1: */* Phase-2: Fine-tuning in a target environment */*
2: The parameters of flow embedding module and allocation function are loaded
3: **while** *The training loss is not converged* **do**
4: The flow embedding module computes flow embeddings by taking flow states from devices
5: The allocation function determines a latent action
6: The adjustment function determines control values
7: The shaping function shapes an action to adapt to the network conditions based on the latent action and control values
8: The training loss is calculated from objective functions
9: The parameter of the adjustment function is updated based on the loss to fine-tune the policy for a target environment
10: **end while**

conditions in a given target environment. This two-phase procedure is detailed in Algorithm 2 and 3.

In Repot, the fine-tuning structure is tailored in that it requires to update only the adjustment function while the other trainable functions are frozen, as shown in the right part in Figure 2. In doing so, we employ the action shaping scheme by which a latent action is first calculated from flow embeddings and then the action is transformed into bandwidth limit values according to specific network conditions.

III. POLICY TRANSFERABLE RL STRUCTURE

In this section, we describe the model structure and algorithms in Repot. The flow embedding module encodes state S^t in Eq. (3) into embeddings, and the bandwidth allocation module calculates action $\mathbf{a}^t$ in Eq. (4) from the embeddings. In the following, we explain those modules in detail.

A. FLOW EMBEDDING

The flow embedding module $\mathrm{EMB}_\psi(\cdot)$ with trainable model parameters ψ consists of vectorization and self-attention functions, and it represents each flow information from a device in a common vector space.

1) VECTORIZATION

For each time-step t, an intermediate embedding vector $\mathbf{e}'_i$ based on a historical flow state $\mathbf{S}_i^t$ is obtained through $\mathrm{MLP}_\psi(\cdot)$, i.e.,

$$\mathbf{e}'_i = \mathrm{MLP}_\psi\left(\mathbf{S}_i^t\right). \tag{6}$$

2) RELATION EXTRACTION

Given intermediate embeddings $\mathbf{E}' = \left[\mathbf{e}'_1, \ldots, \mathbf{e}'_{N_D}\right]$, the respective flow embeddings are obtained by

$$\mathbf{E}^t = \mathrm{ATT}_\psi\left(\mathbf{E}'\right) = \left[\mathbf{e}_1^t, \ldots, \mathbf{e}_{N_D}^t\right]. \tag{7}$$

In $\mathrm{ATT}_\psi(\cdot)$, query, key, and value vectors (i.e., $\mathbf{q}_i$, $\mathbf{k}_i$, and $\mathbf{v}_i$) are calculated through respective MLPs (i.e., $\mathrm{MLP}_{\psi_\mathrm{q}}(\cdot)$, $\mathrm{MLP}_{\psi_\mathrm{k}}(\cdot)$, and $\mathrm{MLP}_{\psi_\mathrm{v}}(\cdot)$). Given each vector in $\mathbf{E}'$, its respective elements are first obtained by

$$x_i = \mathrm{MLP}_{\psi_x}\left(\mathbf{e}'_i\right) \tag{8}$$

for $x \in \{\mathbf{q}, \mathbf{k}, \mathbf{v}\}$. Then, attentive weight vector $\mathbf{w}_i$ representing the relationship between flow state $\mathbf{S}_i^t$ and the others is obtained using the scaled dot-production of $\mathbf{q}_i$ and $\left[\mathbf{k}_1, \ldots, \mathbf{k}_{N_D}\right]$,

$$\begin{aligned}\mathbf{w}_i &= \mathrm{Softmax}\left(\left[\frac{\mathbf{q}_i^\intercal \cdot \mathbf{k}_1}{\sqrt{d}}, \ldots, \frac{\mathbf{q}_i^\intercal \cdot \mathbf{k}_{N_D}}{\sqrt{d}}\right]\right) \\ &= \left[\Pr(w'_1), \ldots, \Pr(w'_{N_D})\right]^\intercal = \left[w_1, \ldots, w_{N_D}\right]^\intercal\end{aligned} \tag{9}$$

where $\Pr(w'_i) = \frac{\exp(w'_i)}{\sum_{j=1}^{N_D}\exp(w'_j)}$. Finally, flow embedding vector $\mathbf{e}_i^t$ is obtained by $(w_1\mathbf{v}_1 + \ldots + w_{N_D}\mathbf{v}_{N_D})$.

As described, flow embeddings encapsulate both historical and relational features of individual flows, and represent those on a common vector space, supporting a scalable model structure in which a policy can be optimized in different network scales. For further explanation, we notate the flow embedding in a simple form combining Eq. (6) and (7).

$$\begin{aligned}\mathbf{E}^t &= \mathrm{EMB}_\psi(S^t) \\ &= \mathrm{ATT}_\psi\left(\left[\mathrm{MLP}_\psi\left(\mathbf{S}_1^t\right), \ldots, \mathrm{MLP}_\psi\left(\mathbf{S}_{N_D}^t\right)\right]\right)\end{aligned} \tag{10}$$

B. BANDWIDTH ALLOCATION

The bandwidth allocation module is structured based on our design principles explained below to provide the adaptation of a learned policy to a target environment. To mitigate a large action space problem [29], [30] and establish fast convergence in model training, we employ the latent action representation. A latent action generated by a trainable function (ALLOC) upon flow embeddings is transformed according to observed network conditions. As the latent action space is much smaller than what is required for individual actions for all sensor devices in terms of dimensions, it can be effective to build the ALLOC function by model training and to use its output for adaptation.

Furthermore, to support the adaptation in a target environment, we restructure the action transformation (that processes the latent action output of ALLOC) into two functions such as a non-trainable, controllable function (SHAPE) that conducts action shaping and a trainable function (ADJUST) that sets the control parameter values of SHAPE. That is, the ADJUST

function is intended to learn to calculate the optimal control values through a small number of training samples, rendering the latent action outputs of ALLOC well-fitted to a target environment.

1) ALLOCATION

A latent action $\tilde{\mathbf{a}}^t = [\tilde{a}^t_1, \ldots, \tilde{a}^t_{N_p}]^\intercal$ is obtained by

$$\tilde{\mathbf{a}}^t = \text{ALLOC}_{\phi_1}(\mathbf{E}^t) \tag{11}$$

where each $\tilde{a}^t$ corresponds to weighted values on a fixed-size set of geographical points $[p_1, \ldots, p_{N_p}]$ for $N_p \ll N_D$. Note that each point is randomly placed on a 2D-grid where sensing devices are located. $\text{ALLOC}_{\phi_1}(\cdot)$ is a ϕ_1-parameterized function trained by RL to induce policy π_{ϕ_1}.

2) ADJUSTMENT

As described in our design principle above, each latent action is transformed into a target-specific action. Specifically, control (parameter) values $\Delta^t = [\tilde{\mathbf{a}}^t_\delta, k^t_\delta, v^t_\delta]$ are first calculated by another policy π_{ϕ_2} with trainable parameters ϕ_2, i.e.,

$$\Delta^t = \text{ADJUST}_{\phi_2}(\mathbf{E}^t). \tag{12}$$

Those control values are used in Eq. (13) below to complete action shaping. To minimize a search space in RL and support the rapid adaptation, we deliberately confine the range of $\tilde{\mathbf{a}}_\delta$ within small $z\%$ of that of $\tilde{\mathbf{a}}$, and have $z = 30\%$ by default.

3) SHAPING

Given control values $[\tilde{\mathbf{a}}^t_\delta, k^t_\delta, v^t_\delta]$ in Eq. (12), a latent action $\tilde{\mathbf{a}}^t$ in Eq. (11) is transformed as

$$\mathbf{a}^t = \text{SHAPE}(\tilde{\mathbf{a}}^t + \tilde{\mathbf{a}}^t_\delta)|_{k+k^t_\delta, v+v^t_\delta} \tag{13}$$

where k and v are constants, having $k = 1$, $v = 0$ by default, and SHAPE is a non-trainable function of which implementation is explained in Eq. (14)-(16) below.

First, intermediate values $\left[a'_1, \ldots, a'_{N_D}\right]$ for bandwidth limits of devices are calculated using the latent action in Eq. (11), the control values in Eq. (12), and the inverse distance from device D_i (for $i \in \{1, \ldots N_D\}$) to a set of geographical points $[p_1, \ldots, p_{N_p}]$, i.e.,

$$\tilde{a}'_i = \sum_{j=1}^{N_P} \frac{\tilde{a}^t_j + \tilde{a}^t_{\delta,j} + \epsilon}{\|D_i - p_j\|^{k+k^t_\delta} + \epsilon}, \tag{14}$$

$$a'_i = \begin{cases} \tilde{a}'_i, & \text{if } \tilde{a}'_i \geq 0.1N_p, \\ c, & \text{otherwise.} \end{cases} \tag{15}$$

Note that $||D_i - p_j||$ is the distance from device D_i to point p_j, and $0.1N_p$ is a clipping threshold, $\epsilon \ll 1$ is a small positive constant, and $c = -2$ is a clipping value.

Then, action $\mathbf{a}^t$ is obtained by transforming the intermediate values calculated above, i.e.,

$$\begin{aligned} \mathbf{a}^t &= (1 - v - v^t_\delta)A \cdot \text{Softmax}\left(\left[a'_1, \ldots, a'_{N_D}\right]\right) \\ &= (1 - v - v^t_\delta)A \cdot \left[\Pr(a'_1), \ldots, \Pr(a'_{N_D})\right]^\intercal \\ &= [a^t_1, \ldots, a^t_{N_D}]^\intercal \end{aligned} \tag{16}$$

where A is the overall bandwidth limit in Eq. (2). Finally, upon receiving action a^t_i about bandwidth allocation at time-step t, sensing device D_i modifies its configuration on the status update rate according to a^t_i.

In Algorithm 4, we combine all the steps involving flow embedding and bandwidth allocation, and represent them in a code snippet with the corresponding equations. The time complexity is $O(d \cdot N_D^2)$ for the embedding dimension d and the number of devices N_D, according to the relation extraction function in the flow embedding module at line 4.

Algorithm 4 Bandwidth Allocation

1: ***// 1. Observation for each historical flow state***
2: Observe a flow state $\mathbf{S}^t_i = \left[\mathbf{s}^t_i, \ldots, \mathbf{s}^{t-u}_i\right]$ in Eq. (3)
3: ***// 2. Flow embedding using the modified self-attention function***
4: Compute flow embeddings $\mathbf{E}^t = \text{EMB}_\psi(S^t)$ in Eq. (10)
5: ***// 3. Establishing a latent action***
6: Determine a latent action $\tilde{\mathbf{a}}^t = \text{ALLOC}_{\phi_1}(\mathbf{E}^t)$ in Eq. (11)
7: ***// 4. Adaptation for a target network environment***
8: Determine control values $[\tilde{\mathbf{a}}^t_\delta, k^t_\delta, v^t_\delta] = \text{ADJUST}_{\phi_2}(\mathbf{E}^t)$ in Eq. (12)
9: ***// 5. Action shaping based on $\tilde{\mathbf{a}}^t$ and $\Delta^t = [\tilde{\mathbf{a}}^t_\delta, k^t_\delta, v^t_\delta]$***
10: Compute an action $\mathbf{a}^t = \text{SHAPE}(\tilde{\mathbf{a}}^t + \tilde{\mathbf{a}}^t_\delta)|_{k+k^t_\delta, v+v^t_\delta}$ in Eq. (13)
11: ***// 6. Bandwidth allocation for each device's flow***
12: Send a packet including the bandwidth limit value a^t_i to device D_i

C. IMPLEMENTATION

To implement the **Repot** model, we use the soft actor-critic (SAC) RL algorithm [23], an off-policy RL method that is known to effectively retain the benefits of entropy maximization and stability. Specifically, we employ two SAC structures, denoted as SAC_m for $m \in \{1, 2\}$. SAC_1 is used to establish a bandwidth allocation policy in a source environment through end-to-end model training, and SAC_2 is used to adapt a learned policy by SAC_1 to a given specific target environment through sample-efficient partial model training (fine-tuning). Accordingly, the dotted line modules in blue in Figure 2 are all trained by SAC_1, and the dotted line module in orange is trained by SAC_2.

In our implementation on SAC_1, the actor includes the model parameters ψ_{Act} and ϕ_1, and the critic includes the model parameters ψ_{Crit} and θ_1. SAC_1 drives all those parameters to be updated in an end-to-end training fashion. Note that θ_1 consists of two soft Q-functions which we represent in $\theta_{1,n}$ for $n \in \{1, 2\}$. The training rollout on SAC_1 is described in Algorithm 5, which corresponds to the detail implementation of Algorithm 2 for (Phase-1) model training.

Unlike SAC_1, SAC_2 is tailored for fine-tuning and adaptation in a target environment. Accordingly, in SAC_2, while the actor includes the model parameters ψ_{Act} and ϕ_2 and the critic includes the model parameters ψ_{Crit} and θ_2, SAC_2 drives only the parameters ϕ_2 and θ_2 to be updated during its training; the other model parameters are fixed. Same as the two soft Q-functions of θ_1, we have $\theta_{2,n}$ for $n \in \{1, 2\}$. The training rollout on SAC_2 is described in Algorithm 6, which

Algorithm 5 Training Rollout in Phase-1

/* *Establishing policy* π_{ϕ_1} *by* SAC_1 *in a source environment* */
// *Training parameter initialization*
Initialize ψ_{Act}, ψ_{Crit}, ϕ_1, α, $\theta_{1,n}$ for $n \in \{1,2\}$
$\bar{\theta}_{1,n} \leftarrow \theta_{1,n}$ for $n \in \{1,2\}$, $\mathcal{D} \leftarrow \emptyset$
// *Model training with total time-step* T *and learning rate* λ
for $t \leftarrow 1$ **to** T **do**
  // *Training sample store*
  for each environment step **do**
    Sample $\tilde{\mathbf{a}}^t \sim \pi_{\phi_1}(\tilde{\mathbf{a}}^t|\text{EMB}_{\psi_{\text{Act}}}(S^t))$,
    Sample $S^{t+1} \sim \rho_{\pi_{\phi_1}}(S^{t+1}|S^t, \tilde{\mathbf{a}}^t)$
    Store $\mathcal{D} \leftarrow \mathcal{D} \cup \{(S^t, \tilde{\mathbf{a}}^t, r^t, S^{t+1})\}$
  end for
  // *Parameter update*
  for each gradient step **do**
    // *Soft Q-function and critic's embedding module update*
    Update $\theta_{1,n} \leftarrow \theta_{1,n} - \lambda\hat{\nabla}_{\theta_{1,n}} J_{\text{Crit}}^{(1,n)}$ for $n \in \{1,2\}$,
      $\psi_{\text{Crit}} \leftarrow \psi_{\text{Crit}} - \lambda\hat{\nabla}_{\psi_{\text{Crit}}}\left(J_{\text{Crit}}^{(1,1)} + J_{\text{Crit}}^{(1,2)}\right)$
    // *Policy and actor's embedding module update*
    Update $\phi_1 \leftarrow \phi_1 - \lambda\hat{\nabla}_{\phi_1} J_{\text{Act}}^{(1)}$, $\psi_{\text{Act}} \leftarrow \psi_{\text{Act}} - \lambda\hat{\nabla}_{\psi_{\text{Act}}} J_{\text{Act}}^{(1)}$
    // *Temperature parameter update*
    Update $\alpha \leftarrow \alpha - \lambda\hat{\nabla}_{\alpha} J_{\alpha}^{(1)}$
    // *Target soft Q-function update*
    Update $\bar{\theta}_{1,n} \leftarrow \chi\theta_{1,n} + (1-\chi)\bar{\theta}_{1,n}$ for $n \in \{1,2\}$
  end for
end for

Algorithm 6 Training Rollout in Phase-2

/* *Establishing policy* π_{ϕ_2} *by* SAC_2 *in a target environment* */
// *Training parameter load and initialization*
Load $\psi_{\text{Crit}}, \psi_{\text{Act}}, \phi_1$, Initialize ϕ_2, α, $\theta_{2,n}$ for $n \in \{1,2\}$
$\bar{\theta}_{2,n} \leftarrow \theta_{2,n}$ for $n \in \{1,2\}$, $\mathcal{D} \leftarrow \emptyset$
// *Fine-tuning with total time-step* T' *and learning rate* λ'
for $t \leftarrow 1$ **to** T' **do**
  // *Training sample store*
  for each environment step **do**
    Sample $\mathbf{\Delta}^t \sim \pi_{\phi_2}(\mathbf{\Delta}^t|\text{EMB}_{\psi_{\text{Act}}}(S^t))|_{\pi_{\phi_1}}$
    Sample $S^{t+1} \sim \rho_{\pi_{\phi_2}}(S^{t+1}|S^t, \mathbf{\Delta}^t)|_{\pi_{\phi_1}}$
    Store $\mathcal{D} \leftarrow \mathcal{D} \cup \{(S^t, \mathbf{\Delta}^t, r^t, S^{t+1})\}$
  end for
  // *Parameter update*
  for each gradient step **do**
    // *Soft Q-function update*
    Update $\theta_{2,n} \leftarrow \theta_{2,n} - \lambda'\hat{\nabla}_{\theta_{2,n}} J_{\text{Crit}}^{(2,n)}|_{\pi_{\phi_1}}$ for $n \in \{1,2\}$
    // *Adaptation policy update*
    Update $\phi_2 \leftarrow \phi_2 - \lambda'\hat{\nabla}_{\phi_2} J_{\text{Act}}^{(2)}|_{\pi_{\phi_1}}$
    // *Temperature parameter update*
    Update $\alpha \leftarrow \alpha - \lambda'\hat{\nabla}_{\alpha} J_{\alpha}^{(2)}|_{\pi_{\phi_1}}$
    // *Target soft Q-function update*
    Update $\bar{\theta}_{2,n} \leftarrow \chi\theta_{2,n} + (1-\chi)\bar{\theta}_{2,n}$ for $n \in \{1,2\}$
  end for
end for

corresponds to the detail implementation of Algorithm 3 for (Phase-2) fine-tuning.

In the following, we describe several objective functions used to train SAC_m, where we use the index notation $m, n \in \{1,2\}$ to represent each of the two SAC structures (SAC_m) and the two Q-functions ($Q_{\theta_{m,n}}(\cdot)$) for each SAC structure explained above, respectively. We also represent a ϕ_m-parameterized policy as π_{ϕ_m}, and replace $\tilde{\mathbf{a}}^t$ with Δ^t for SAC_2.

1) CRITIC

We optimize the soft Q-functions $Q_{\theta_{m,n}}(\cdot)$ and the flow embedding module $\text{EMB}_{\psi_{\text{Crit}}}(\cdot)$ in Eq. (10) with the parameters ψ_{Crit} for policy π_{ϕ_m} by minimizing the soft Bellman residual,

$$J_{\text{Crit}}^{(m,n)} = \mathop{\mathbb{E}}_{(S^t,\tilde{\mathbf{a}}^t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_{\theta_{m,n}}(\text{EMB}_{\psi_{\text{Crit}}}(S^t), \tilde{\mathbf{a}}^t) - \left(r^t + \gamma\mathbb{E}_{S^{t+1}\sim\rho_{\pi_{\phi m}}}[V_{\bar{\theta}_{m,n}}(\mathbf{E}_{\text{Crit}}^{t+1})]\right)\right)^2\right] \quad (17)$$

where

$$V_{\bar{\theta}_{m,n}}(\mathbf{E}_{\text{Crit}}^{t+1}) = \mathop{\mathbb{E}}_{\tilde{\mathbf{a}}^{t+1}\sim\pi_{\phi m}}\left[Q_{\bar{\theta}_{m,n}}(\text{EMB}_{\psi_{\text{Crit}}}(S^{t+1}), \tilde{\mathbf{a}}^{t+1}) - \alpha\log\pi_{\phi_m}(\tilde{\mathbf{a}}^{t+1}|\text{EMB}_{\psi_{\text{Act}}}(S^{t+1}))\right]. \quad (18)$$

Here, $\mathcal{D}$ is the replay pool, γ is the discount factor, $\rho_{\pi_{\phi m}}$ is the marginals of trajectory distribution induced by policy π_{ϕ_m}, and α is the adjustable temperature parameter that controls the stochasticity of the optimal policy [23]. $\bar{\theta}_{m,n}$ denotes the parameter of a target soft Q-function, obtained as an exponentially moving average of the soft Q-function weights.

2) ACTOR

We optimize policy π_{ϕ_m} and the flow embedding module $\text{EMB}_{\psi_{\text{Act}}}(\cdot)$ in Eq. (10) with the parameters ψ_{Act} by minimizing the below objective function,

$$J_{\text{Act}}^{(m)} = \mathop{\mathbb{E}}_{\substack{S^t\sim\mathcal{D}\\ \epsilon^t\sim\mathcal{N}}}[\alpha\log\pi_{\phi_m}(f_{\phi_m}(\epsilon^t; \text{EMB}_{\psi_{\text{Act}}}(S^t))|\text{EMB}_{\psi_{\text{Act}}}(S^t)) - Q_{\theta_m}^{\text{Min}}(\text{EMB}_{\psi_{\text{Crit}}}(S^t), f_{\phi_m}(\epsilon^t; \text{EMB}_{\psi_{\text{Act}}}(S^t)))] \quad (19)$$

where $f_{\phi_m}(\epsilon^t; \text{EMB}_{\psi_{\text{Act}}}(S^t)) = \tilde{\mathbf{a}}^t$ is the neural network transformation to re-parameterize the policy, ϵ^t is a noise vector sampled from a Gaussian, and $Q_{\theta_m}^{\text{Min}}(\cdot)$ is used as the minimum of the soft Q-functions for policy gradient.

3) TEMPERATURE PARAMETER

To improve the performance and stability of the SAC algorithm, we use the following objective to calculate gradients for temperature parameter α,

$$J_{\alpha}^{(m)} = \mathop{\mathbb{E}}_{\tilde{\mathbf{a}}^t\sim\pi_{\phi_m}}[-\alpha\log\pi_{\phi_m}(\tilde{\mathbf{a}}^t|\text{EMB}_{\psi_{\text{Act}}}(S^t)) - \alpha\bar{\mathcal{H}}] \quad (20)$$

where $\bar{\mathcal{H}}$ denotes the desired minimum entropy [23].

IV. EVALUATION

In this section, we describe the implementation of Repot, and evaluate its performance compared to other algorithms including a learning model based on a state-of-the-art AoI algorithm [3] under various network conditions. We also provide our case study with a microscopic multi-modal urban mobility simulator.

A. LEARNING ENVIRONMENTS

In evaluation, we consider a networked monitoring system where status updates of sensing devices are aggregated for continual query processing. We built a simulation environment for such networked monitoring based on NS-3 [25] v3.29, a widely used packet-level discrete-event network simulator, and we use the ZeroMQ messaging library [31] for asynchronous communication between the NS-3 based simulation environment and the modules in Repot.

In networked monitoring, we focus on data-driven real-time queries whose quality is dependent on the timeliness of status updates. For example, a real-time traffic map is constructed using low-latency image streams generated from geographically distributed camera devices, where traffic congestion can be estimated for autonomous vehicle navigation [32]. This networked monitoring capability is required in a variety of data-driven applications, e.g., urban air quality inference [33], target tracking [34].

To evaluate the quality of query processing through QUAL(·) in Eq. (2), we use structural similarity index measure (SSIM) [35] that estimates the similarity between the aggregated status information I^τ (at the server) and the ground truth I^τ_{Truth} (at the sensing devices) at time τ within a time-step interval.

$$\text{QUAL}(I^t) \approx \sum_\tau \text{SSIM}(x, y) \tag{21}$$

where

$$\text{SSIM}(x, y) = \frac{2\mu_x\mu_y + \epsilon_1}{\mu_x^2\mu_y^2 + \epsilon_1} + \frac{2\sigma_x\sigma_y + \epsilon_2}{\sigma_x^2\sigma_y^2 + \epsilon_2} + \frac{\sigma_{xy} + \epsilon_3}{\sigma_x^2\sigma_y^2 + \epsilon_3}. \tag{22}$$

Note that μ and σ are the mean and standard deviation of $x = I^\tau$ and $y = I^\tau_{\text{Truth}}$, and $\epsilon_j \ll 1$ are small positive constants for $j \in \{1, 2, 3\}$. QUAL(I^t) is evaluated several times with a uniform-random interval of [6.7, 8.3] within the time-step interval of 33.3ms.

In our simulation tests, we construct two different wireless network environments. Table 3 illustrates the environment settings where the source $\mathcal{B}$ denotes a learning environment and the target $\mathcal{T}$ denotes an adaptation and testing environment. With these source and target environments, our experiment aims at verifying the policy transferability in Repot across different network conditions.

The source $\mathcal{B}$ is set to have easy-to-learn network conditions characterized as ideal time-slotted communications [36] with no channel access collision, no propagation loss, and no other errors in transmission; yet given a resource constraint on link capacity, channel access delay and propagation delay are modeled. The target $\mathcal{T}$ is set to have complex network conditions similar to real-world deployment conditions, and it is implemented using several modules in NS-3, e.g., WiFi, network, internet, mobility modules including NistErrorRateModel, ConstantSpeedDelayModel, and LogDistanceLossModel.

TABLE 3. Experiment network conditions for source environment $\mathcal{B}$ and target environment $\mathcal{T}$: The default settings for $\mathcal{T}$ are in parentheses.

Config.	Wireless network env.	
	Source env. $\mathcal{B}$	Target env. $\mathcal{T}$
MAC protocol	Ideal slotted aloha	802.11ac
VHT MCS	-	8, 7, 6, 5 (6)
Link capacity	357.0 Mbps	702, 585, 526.5, 468 (526.5) Mbps
Channel bandwidth	-	80 MHz
Num. of antennas	-	2
Initial backoff window	-	64
Backoff stage count	-	5
Data/control packet size	1518 Bytes	
Transport protocol	UDP	
Information frame size	16 KBytes	
Total bandwidth limit	357.0 Mbps	

TABLE 4. Training sample generation time (days) in source and target environments: This was measured on a system of an Intel(R) Core(TM) i9-10900x processor and an NVIDIA RTX 3090 GPU.

Env.	Days for 1.6M time-steps				
	Num. of sensing devices				
	49	64	100	144	256
Source	0.2	0.3	0.5	0.6	1.0
Target	36.0	55.2	88.6	131.5	274.2

Table 4 illustrates the time in days required to generate training samples, i.e., RL states during 1.6M time-steps, which are used to achieve a converged model in our source environment. In the NS-3 based target environment, model training would take at least 36 days even for a system of 49 sensing devices, but in the source environment, it takes less than 5 hours. This difference in training times indicates the benefits of policy transferability across different environments. It is desirable to have a policy learned in an easy-to-learn source environment within a reasonable training time and to adapt the policy rapidly to target environments.

B. COMPARISON METHODS

For model training, we use SAC modules in the Stable-baselines3 [37] and PyTorch [38] v1.5.1. In addition to our Repot, we test several heuristic- and learning-based methods including ARN-RL that exploits a state-of-the-art AoI-centric algorithm [3]. Each method below determines the bandwidth limit (or the transmission rate) a_i^t of device D_i at time-step t.

- **Uniform.** All devices have an equally distributed bandwidth limit a^t. This method is used to set the reference performance for comparison.
- **Random.** Each device is assigned a randomly distributed a^t at every time-step.
- **Top-Opt.** K in Top-K is set to be optimal for a given environment. In Top-K, the top $K\%$ ranked devices share a common bandwidth margin, e.g., Top-20 allows the top 20% ranked devices to have $\frac{A}{20}$ where A is the overall bandwidth limit. The ranking score of device D_i is calculated based on the total number of objects captured by those devices close to D_i within range r, i.e.,

$$\text{SCORE}(D_i) = \sum_{j=1}^{N_D} \text{OB}(I_j^t) \quad \text{s.t. } ||D_i - D_j|| \leq r \quad (23)$$

  where $\text{OB}(\cdot)$ yields the number of objects observed by D_j in I_j^t, and $||D_i - D_j||$ denotes the distance of D_i and D_j.
- **Naïve-RL.** An RL model is trained by the actor-critic policy gradient method to set individual a_i^t. The actor and critic are implemented with each five-layer MLP.
- **ARN-RL.** A model with the attention-integrated relevance network (ARN) [3] is implemented to make use of a state-of-the-art AoI-centric scheme in networked monitoring. ARN is intended to extract important features from the observed states and previous executed action. In our implementation, each module of three feed-forward layers for actor's policy function and critic's Q-functions takes those features as input to determine the next action.

The hyperparameter settings for the RL-based methods aforementioned such as Naïve-RL and ARN-RL, as well as our Repot model are summarized in Table 5.

TABLE 5. Hyperparameter settings for RL-based methods.

Hyperparameter	Value
Optimizer	Adam
Discount factor γ	0.9
Normal training time-steps T	1,600,000
Adaptation training time-steps T'	100,000
Target smoothing coef.	0.005
Normal learning rate λ	0.00005
Adaptation learning rate λ'	0.0005
Training frequency (in a time-step)	1
Gradient time-steps	1
Replay buffer size	800,000
Mini-batch size	256
Num. of points N_P	4, 5, 6

C. QoE PERFORMANCE

Using the SSIM-based quality in Eq. (21), we evaluate Repot, compared to the other methods. We measure the ratio of achieved SSIM to an ideal reference. That is, for a method $\mathcal{M}$, the relative QoE is calculated as

$$\text{Relative QoE } (\%) = \frac{\text{QoE}_{\mathcal{M}}}{\text{QoE}_{\text{Uniform}}} \times 100 \quad (24)$$

where $\text{QoE}_{\text{Uniform}}$ denotes the ideal reference QoE and $\text{QoE}_{\mathcal{M}}$ denotes the achieved QoE by the method $\mathcal{M}$. QoE is estimated by the average quality QUAL in Eq. (2). $\text{QoE}_{\text{Uniform}}$ is calculated based on the QoE achieved by the Uniform method in the reference network environment that is intentionally built to measure the ideal reference performance; unlike the source $\mathcal{B}$ and target $\mathcal{T}$, this reference network is set to have neither channel access delay nor propagation delay.

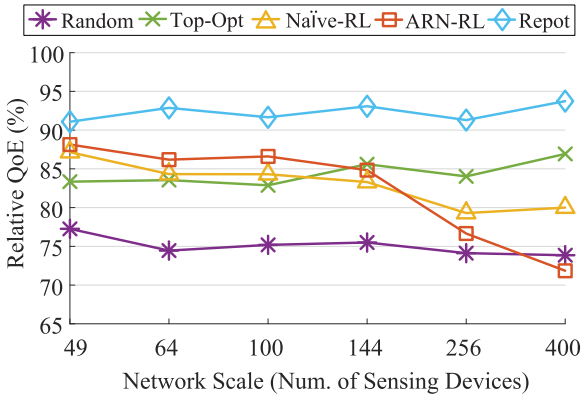


FIGURE 3. QoE in the source $\mathcal{B}$ with respect to network scales: The x-axis denotes the number of sensing devices in a networked monitoring system and the y-axis denotes the achieved performance by compared methods in relative QoE in Eq. (24).

Figure 3 represents the performance of Repot and the other methods which are trained and tested in $\mathcal{B}$ with respect to various network scales. As shown, Repot outperforms the other methods for all the cases, maintaining high relative QoE of 91∼94% while the other methods show lower relative QoE less than 90%. Repot and the heuristic-based methods (Random and Top-Opt) maintain stable performance in relative QoE regardless of scales, while Naïve-RL and ARN-RL are affected in a large scale. For example, ARN-RL shows better performance than the others except for Repot at the sizes of 49, 64, and 100, but it shows performance degradation at the sizes of 256 and 400. Naïve-RL also has a similar pattern, showing 7.1% degradation from 49 to 400. This policy deterioration has been discussed in the RL context of the curse of dimensionality [39] and large action space [29].

In contrast, Repot maintains high relative QoE at the sizes of 256 and 400. In Repot, a policy is learned in a low latent space and then a latent action is mapped to a target-dependent large action space through the action shaping scheme. This latent action structure renders RL models scalable to a large action space which is common in networked, sensor-based monitoring systems, along with the flow embedding that abstracts complex network states in a common vector space of a fixed size.

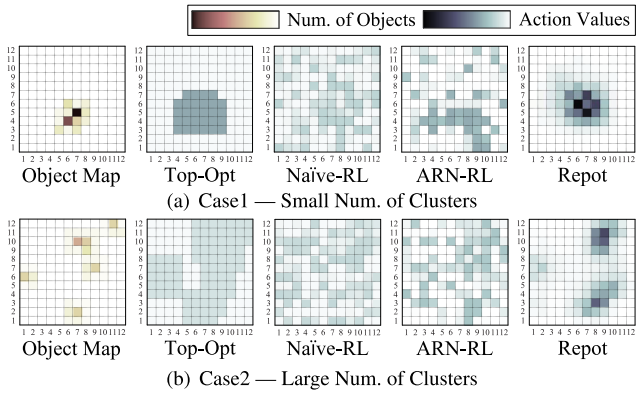


FIGURE 4. Bandwidth allocation patterns by different methods: The first column describes object maps in a 2-dimensional geographical space, where a darker cell denotes more objects within it. The other columns describe bandwidth allocation patterns by methods where a darker cell denotes higher bandwidth limits (action values) assigned to the devices close to the cell. The two cases in (a) and (b) represent the object maps with different numbers of object clusters. Each grid cell (a small square in the figure) is associated with x- and y-coordinates that represent its spatial feature in the geographical space.

In Figure 4, we illustrate the bandwidth allocation patterns by different methods. The x- and y-coordinates indicate the location of sensor devices on a geographical space where each device in a grid cell (a square in this figure) sends the real-time information about its observation within the cell to a server. Interestingly, **Naïve-RL** shows such a tendency to cover the entire area with a slight concentration on object locations. On the contrary, both **ARN-RL** and **Repot** have more focus on object locations, showing the effect of attention mechanism. However, while both use attention, unlike **ARN-RL** which extracts action directly from a learned policy, **Repot** employs the latent action and action shaping which lead to fine-grained action representation that can be appropriately formed to observed state changes over time. For example, **Repot** shows more relevant patterns than the others for the number of clusters in the object maps. Interestingly, the patterns of **Top-Opt** and **Repot** have in common that they make concentration on dense areas or clusters. However, **Top-Opt** considers only current dense areas but **Repot** tends to consider their trajectory.

D. ADAPTATION PERFORMANCE

As it is expensive to establish training samples sufficiently about networked monitoring systems in various environment conditions, we discuss the policy transferability of **Repot** across different conditions. We explore policy transferability by evaluating how well methods adapt to a target environment in a small number of training samples, e.g., within 100K time-steps and 6.25% of 1.6M in Table 4.

For adaptation in a target environment $\mathcal{T}$, we train the models of **Naïve-RL**, **ARN-RL**, and **Repot** in the source environment $\mathcal{B}$ with the dynamic link capacity of $\left[\frac{1}{2}L_{\mathcal{B}}, L_{\mathcal{B}}\right]$, where $L_{\mathcal{B}}$ is the link capacity of $\mathcal{B}$. Then, we fine-tune each learned model (policy) $\mathcal{M}$ in $\mathcal{T}$ through top-layer updates in conventional transfer learning [40], [41] for **Naïve-RL** and **ARN-RL**. For **Repot**, we employ fine-tuning with the action shaping, in which only $\text{ADJUST}_{\phi_2}(\cdot)$ in Eq. (12) is updated. In addition, we update the K value of **Top-K** to be optimized in $\mathcal{T}$ for **Top-Opt**.

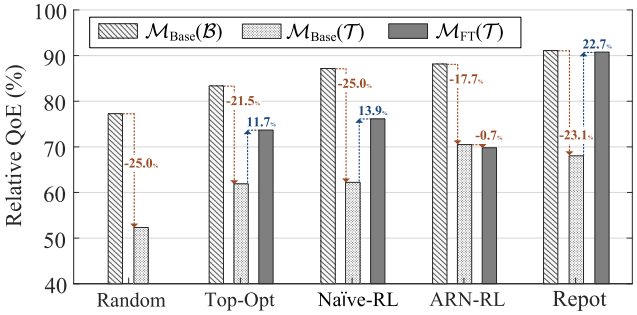


FIGURE 5. Adaptation performance in a target environment: For each method in the x-axis, $\mathcal{M}_{\text{Base}}(\mathcal{B})$ denotes the performance in the source environment $\mathcal{B}$, $\mathcal{M}_{\text{Base}}(\mathcal{T})$ denotes the performance in the target environment $\mathcal{T}$ before fine-tuning, and $\mathcal{M}_{\text{FT}}(\mathcal{T})$ denotes the performance in $\mathcal{T}$ after fine-tuning.

Figure 5 shows the adaptation performance of methods. For each method, applying its policy optimized in $\mathcal{B}$ incurs significant degradation in $\mathcal{T}$, e.g., 21% degradation by **Naïve-RL** (from $\mathcal{M}_{\text{Base}}(\mathcal{B})$ to $\mathcal{M}_{\text{Base}}(\mathcal{T})$). After fine-tuning, some of the methods show stable recovery to some extent. Most importantly, **Repot** shows superior resilience upon the change from $\mathcal{B}$ to $\mathcal{T}$, achieving the recovery of 22.7%, and outperforming the others with 14.5∼20.8% higher relative QoE in $\mathcal{T}$. Specifically, in terms of relative QoE, **Repot** achieves 14.5% higher performance compared to **Naïve-RL** and 20.8% higher compared to **ARN-RL** in the target environment $\mathcal{T}$.

This performance achieved in $\mathcal{T}$ by **Repot** is not only better than that of the other methods but also comparable to that in $\mathcal{B}$ by itself (i.e., 91.0% in $\mathcal{M}_{\text{Base}}(\mathcal{B})$ and 90.7% in $\mathcal{M}_{\text{FT}}(\mathcal{T})$; their difference is no more than 1%). This result clarifies the policy transferability of the modular structure in **Repot** tailored for bandwidth allocation strategies in different network environments. Notice that **ARN-RL** in $\mathcal{T}$ shows a different pattern, having no performance recovery after fine-tuning. We speculate that conventional fine-tuning methods with layer-wise parameter updates are hardly effective to adapt a model optimized in a specific environment to another target, unless a learned policy in a source environment can overfit less and fine-tuning in a target environment can have sufficient samples to overwrite the model parameters.

Figure 6 compares the adaptation efficiency of RL-based methods, where the learning graphs over time-steps correspond to what we have for fine-tuning in Figure 5. Notice that the learning efficiency of $\textbf{Repot}_{\text{FT}}$ is attributed to the action shaping in which a learned policy is adjusted through control value updates that require fewer learning steps. We observe the low performance for $\textbf{ARN-RL}_{\text{FT}}$ and $\textbf{Repot}_{\text{FT}}$ during the first period of learning steps due to our learning with

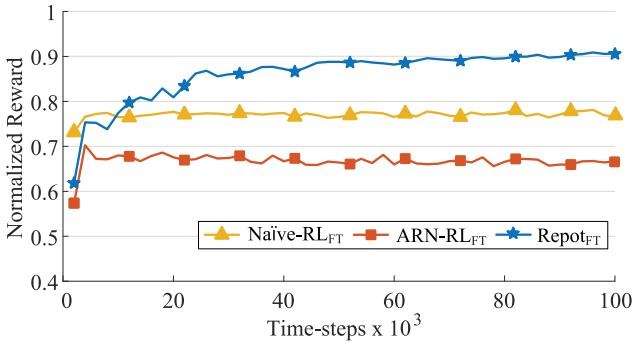


FIGURE 6. Learning graph in adaptation: Naïve-RL$_{\mathsf{FT}}$ denotes the learning graph of adaptation for Naïve-RL in Figure 5. The others denote the respective learning graphs.

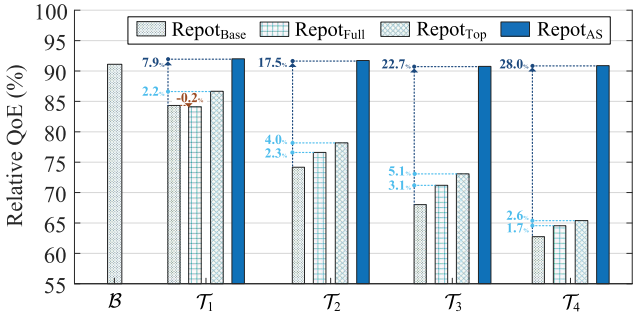


FIGURE 7. Adaptation performance of Repot in various target environments: The target environments $\mathcal{T}_1$∼$\mathcal{T}_4$ are configured differently by VHT MCS settings that determine the wireless data rates. $\mathcal{T}_1$, $\mathcal{T}_2$, and others correspond to VHT MCS 8, 7 and so on. We compare Repot models with different fine-tuning approaches in terms of adaptation performance.

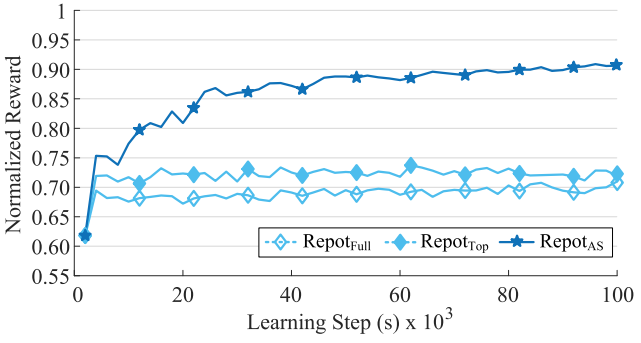


FIGURE 8. Learning graph of different fine-tuning schemes: The learning graphs correspond to the fine-tuning process for the target environment $\mathcal{T}_3$ by different Repot models (with different fine-tuning approaches) in Figure 7.

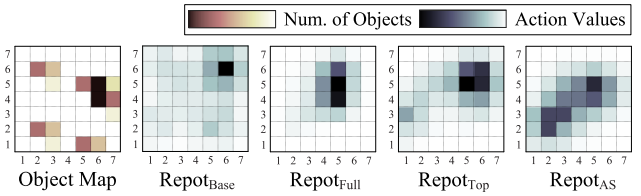


FIGURE 9. Bandwidth allocation actions differently shaped by the fine-tuning approaches in Repot: The meanings of grid cells, grid cell colors and grid cell space are the same as those in Figure 4.

domain randomization. Furthermore, the limited performance improvement of Naïve-RL$_{\mathrm{FT}}$ and ARN-RL$_{\mathrm{FT}}$ specifies the restriction of conventional fine-tuning with partial parameter updates, particularly when environment conditions can vary significantly.

In Figure 7, we evaluate the adaptation of **Repot** across different network environments. We deliberately set the VHT MCS setting of 802.11ac to simulate different target environments $\mathcal{T}_1$∼$\mathcal{T}_4$ with various theoretical link capacities, where the smaller the VHT MCS, the lower the capacity and QoE performance. In this experiment, we employ different fine-tuning approaches in **Repot** to evaluate the policy transferability by action shaping in the same model structure. In **Repot**$_{\mathrm{Base}}$, we use the base model optimized in $\mathcal{B}$ without fine-tuning. In **Repot**$_{\mathrm{AS}}$, we use our proposed action shaping, while we use other fine-tuning schemes in **Repot**$_{\mathrm{Top}}$ and **Repot**$_{\mathrm{Full}}$. Specifically, **Repot**$_{\mathrm{Top}}$ updates the last layer parameters in the ALLOC function, similar to conventional transfer learning, and **Repot**$_{\mathrm{Full}}$ updates its full layer parameters for fine-tuning, considering significant differences between source and target environments.

Overall, **Repot**$_{\mathrm{AS}}$ achieves robust performance across all $\mathcal{T}_1$∼$\mathcal{T}_4$, showing highly comparable adaptation performance in $\mathcal{T}_1$∼$\mathcal{T}_4$ to the respective one in $\mathcal{B}$. The other fine-tuned models **Repot**$_{\mathrm{Top}}$ and **Repot**$_{\mathrm{Full}}$ show limited adaptability, which is consistent with the learning graphs in Figure 8. Furthermore, in Figure 9, we visualize the action patterns of **Repot** differently shaped by the fine-tuning approaches, where the action value represents how much bandwidth is allocated. The action values by **Repot**$_{\mathrm{AS}}$ are much different from those of **Repot**$_{\mathrm{Base}}$ than the others, and they often tend to spread. We speculate that **Repot**$_{\mathrm{AS}}$ is able to properly adapt its policy to target environments that involve uncertainty more than our source environment due to realistic network settings.

E. CASE STUDY

In the following, we show our case study for a networked monitoring system with urban- and suburb-scale car traffic datasets that are generated by the SUMO simulator [24] on the maps of Midtown Manhattan in New York and a suburb of Orlando, Florida. We make the datasets publicly available on Github [42]. Table 6 describes the configuration to generate the datasets, where the urban-scale datasets have about 10 times heavier traffic than the suburb-scale datasets.

TABLE 6. Dataset characteristics.

Config.	Urban-scale	Suburb-scale
Map size	10.1 km^2	5.9 km^2
GPS coordinates	40.76, -73.98	28.62, -81.09
Maximum num. of car	200	46
Average num. of car	112	17
Car moving speed	[1.38, 83.33]m/s	

TABLE 7. Summary of related studies.

Area	Researcher	Target environments	Objective/Application	Approach	Learning structure	Metric
RL in wireless networks	Santan *et al.* [43]	Coexisted WiFi and LTE-U networks	Channel management	LTE-U duty-cycle control	Q-network	Aggregated throughput
	Bhattacharyya *et al.* [20]	Wireless edge networks	Video streaming	Transmission queue priority reconfiguration for buffer control	Q-network	QoE for video streaming
	Elgabli *et al.* [44]	URLLC networks	Timely information update	Status update scheduling for sensor devices	Actor-critic network	Age of information
	Abd-Elmagid *et al.* [45]	UAV-assisted networks	Timely information update	Status update scheduling for sensor devices	Q-network	Age of information
	Traub *et al.* [3]	Wireless sensor networks	Timely information update	Application-oriented status update scheduling	Q-network	Age of correlated information
Domain adaptation	Tobin *et al.* [46]	Robot arms in real-world	Object location detection	Training using randomized manifold data	Convolutional networks	Location detection error
	Peng *et al.* [47]	Robot arms in real-world	Object pushing	Training in simulation with dynamic settings	Policy and value networks	Success rate for pushing objects
	Losey *et al.* [48], [49]	Robot arms in real-world	Remote assistive arm control	Action embedding	Variational autoencoder	Success rate for picking up objects

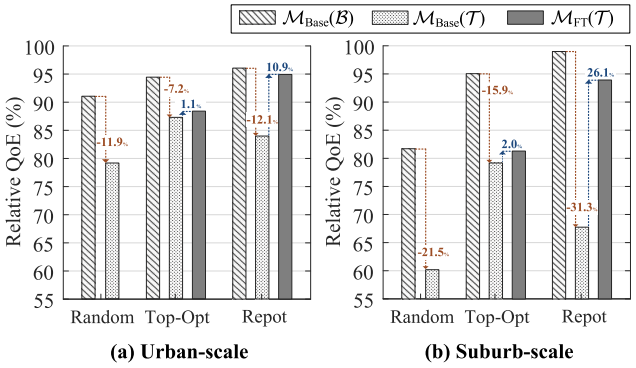


FIGURE 10. Performance comparison in real traffic scenarios: The case study is conducted with two car traffic datasets. The figure on the left shows the performance in (a) urban-scale dataset generated on the map of Midtown Manhattan in New York, and the figure on the right corresponds to the performance in (b) suburb-scale dataset generated on the map of a suburb of Orlando in Florida.

In this case study, we set the underlying network conditions of source and target environments the same as in the previous tests in Table 3.

Figure 10 shows the performance of several methods in the case study, urban-scale in (a) and suburb-scale in (b). We conduct fine-tuning the **Repot** model achieved in the source $\mathcal{B}$ for the target $\mathcal{T}$ using 6.25% of the training sample amount used in $\mathcal{B}$. The **Repot** model yields the best performance (i.e., 96.1% in (a) and 99.6% in (b)) in $\mathcal{B}$. More importantly, the fine-tuned **Repot** model yields the best performance (i.e., 94.9% in (a) and 93.9% in (b)) in $\mathcal{T}$. Whereas the **Repot** model before fine-tuning in $\mathcal{T}$ yields relatively lower performance (i.e., 84.0% in (a) and 67.7% in (b)), its performance recovery after fine-tuning is significant (i.e., 10.9% in (a) and 26.1% in (b)). **Repot** shows higher quality than **Top-Opt** (i.e, 6.5% in (a) and 12.6% in (b)). This demonstrates the capability of **Repot** to learn a near-optimal policy for a given environment through model training and fine-tuning.

There is a larger difference before and after fine-tuning in (b) than in (a). The datasets of the suburb-scale scenario are more cluster-centric as they have less traffic, while the datasets of the urban-scale scenario are less cluster-centric as they have much heavier traffic on the entire region. Such dataset difference provides more margin for **Repot** to be optimized on the dataset of the suburb-scale scenario.

V. RELATED WORK

In network monitoring systems, information quality has been discussed in the context of AoI [4], and several heuristic solutions to AoI problem settings have been introduced; they considered specific network conditions such as unreliable broadcast channels [50], throughput constraints [51], channel interference constraints [52], and dynamic channel status [5].

Recently, RL has been leveraged to address optimization problems upon time-varying network conditions in different problem settings such as network traffic control [53]–[55], wireless channel management [43], [56], and bandwidth allocation for video streaming [20], [57], [58]. Several research works have demonstrated the applicability of RL for AoI problems in networked monitoring systems [3], [44], [45]. Elgabli *et al.* [44] presented an RL-based resource scheduling algorithm to orchestrate sensors and optimize the expected AoI, satisfying the requirement of ultra-reliable low latency communication (URLLC). Abd-Elmagid *et al.* [45] explored the RL-based scheduling on information transmission in unmanned aerial vehicle (UAV)-assisted networks, showing the capability of RL to improve QoE when features are well-defined to learn the flight trajectory of UAVs and the energy consumption pattern of sensors. Similarly, Traub *et al.* [3] addressed the transmission scheduling problem by exploiting application-specific features and attention mechanism.

Those prior works exploited RL and focused on QoE or AoI enhancement, but they rarely addressed the issue of RL model training and fine-tuning adaptability for different

network conditions. Our work also employs RL to improve QoE in networked monitoring systems. However, unlike the prior works, our work concentrates on the policy transferability in RL, which enables the adaptation of a learned policy to different target network environments.

In the field of robotics, model adaptation schemes have been investigated, aiming at bridging the data mismatch gap between simulation environments and real-world robot deployed environments. Tobin *et al*. [46] exploited domain randomization with manifold data for RL-based object detectors. Peng *et al*. [47] developed an RL-based robot arm controller operating in dynamic environments.

To mitigate the problem of sample-inefficiency and large learning time in domain randomization with manifold data, several studies were recently introduced such as building realistic training data [59]–[61] and robot action embedding [48], [49]. Particularly, Losey *et al*. [48], [49] explored the concept of action embedding [29], [30] to improve the control performance of remote assistive robots, which is similar to our action shaping scheme. Whereas the robot action embedding in [48], [49] requires target-specific training data during model training, in our work, a policy is established independently from targets, and an action by a learned policy can be shaped for any target later with a small amount of target-specific training samples.

To the best of our knowledge, our work is the first to discuss RL model adaptation in the context of networked monitoring applications and investigate a modular structure of RL models to provide fast adaptation in different network conditions. Table 7 provides a summary of the related studies.

VI. CONCLUSION

In this paper, we proposed **Repot**, the transferable RL model that enables the QoE enhancement in networked monitoring systems and the efficient adaptation to various target network conditions. In doing so, we employ flow embedding and action shaping by which a bandwidth allocation policy for QoE-driven networked monitoring is trained in an easy-to-learn source environment, and an action by a learned policy can be shaped to target conditions. Through simulation and experiments, we demonstrate that **Repot** achieves competitive QoE performance, outperforming other methods in many cases for both source and target environments. For example, **Repot** achieves high quality in networked monitoring of 14.5∼20.8% gains over the compared methods in a target environment, by fine-tuning with only 6.25% of the samples that are originally required for model training from scratch.

Our direction to future works is to adapt meta RL and multi-task learning for adaptation against different network conditions as well as a variety of application-specific, network-related tasks such as traffic engineering, caching, routing, or intrusion detection. We are also interested in the real-world deployment and testing of transferable RL for AI-based surveillance applications that are required to provide low-latency and high-accuracy model inference, despite harsh network environments.

REFERENCES

[1] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon, "On the role of age of information in the Internet of Things," *IEEE Commun. Mag.*, vol. 57, no. 12, pp. 72–77, 2019.
[2] Z. Jiang, S. Fu, S. Zhou, Z. Niu, S. Zhang, and S. Xu, "AI-assisted low information latency wireless networking," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 108–115, Feb. 2020.
[3] B. Yin, S. Zhang, and Y. Cheng, "Application-oriented scheduling for optimizing the age of correlated information: A deep-reinforcement-learning-based approach," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8748–8759, Sep. 2020.
[4] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 2731–2735.
[5] Z. Jiang, S. Zhou, Z. Niu, and C. Yu, "A unified sampling and scheduling approach for status update in multiaccess wireless networks," in *Proc. IEEE Conf. Comput. Commun.*, Paris, France, Apr. 2019, pp. 208–216.
[6] M. K. Banavar, A. D. Smith, C. Tepedelenlioglu, and A. Spanias, "On the effectiveness of multiple antennas in distributed detection over fading MACs," *IEEE Trans. Wireless Commun.*, vol. 11, no. 5, pp. 1744–1752, May 2012.
[7] D. Ciuonzo, G. Romano, and P. S. Rossi, "Channel-aware decision fusion in distributed MIMO wireless sensor networks: Decode-and-fuse vs. Decode-then-fuse," *IEEE Trans. Wireless Commun.*, vol. 11, no. 8, pp. 2976–2985, Aug. 2012.
[8] D. Ciuonzo, P. S. Rossi, and S. Dey, "Massive MIMO channel-aware decision fusion," *IEEE Trans. Signal Process.*, vol. 63, no. 3, pp. 604–619, Feb. 2015.
[9] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proc. 30th Int. Conf. Very Large Data Bases*, Toronto, ON, Canada, Aug. 2004, pp. 588–599.
[10] J. Traub, S. Breß, T. Rabl, A. Katsifodimos, and V. Markl, "Optimized on-demand data streaming from sensor nodes," in *Proc. 2017 ACM Symp. Cloud Comput.*, Santa Clara, CA, USA, Sep. 2017, pp. 586–597.
[11] L. Hu, Z. Chen, Y. Dong, Y. Jia, L. Liang, and M. Wang, "Status update in IoT networks: Age-of-information violation probability and optimal update rate," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11329–11344, Jul. 2021.
[12] P. Whittle, "Restless bandits: Activity allocation in a changing world," *J. Appl. Probab.*, vol. 25, pp. 287–298, Jan. 1988.
[13] S. K. Sengupta and S. M. Kay, "Fundamentals of statistical signal processing: Estimation theory," *Technometrics*, vol. 37, no. 4, p. 465, Nov. 1995.
[14] S. M. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.
[15] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud, Grid Comput.*, Melbourne, VIC, Australia, May 2010, pp. 826–831.
[16] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Math. Comput. Model.*, vol. 58, no. 5, pp. 1222–1235, 2013.
[17] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, Atlanta, GA, USA, Nov. 2016, pp. 50–56.
[18] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Interest Group Data Commun. (SIGCOMM)*, Beijing, China, Aug. 2019, pp. 270–288.
[19] M. Cheong, H. Lee, I. Yeom, and H. Woo, "SCARL: Attentive reinforcement learning-based scheduling in a multi-resource heterogeneous cluster," *IEEE Access*, vol. 7, pp. 153432–153444, 2019.
[20] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, R. K. P. Mok, and A. Dhamdhere, "QFlow: A reinforcement learning approach to high QoE video streaming over wireless networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Catania, Italy, Jul. 2019, pp. 251–260.
[21] R. Li, Z. Zhao, Q. Sun, C. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.
[22] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive network management in sliced 5G networks with deep learning," in *Proc. IEEE Conf. Comput. Commun.*, Paris, France, Apr. 2019, pp. 280–288.

[23] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[24] D. Krajzewicz, G. Hertkorn, P. Peter Wagner, and C. Rössel, "SUMO (simulation of urban mobility): An open-source traffic simulation," in *Proc. 4th Middle East Symp. Simulation Modeling*, Berlin, Germany, Sep. 2002, pp. 183–187.

[25] *NS-3*. Accessed: Aug. 6, 2021. [Online]. Available: https://github.com/nsnam/ns-3-dev-git.git

[26] M. Kranz, P. Holleis, and A. Schmidt, "Embedded Interaction: Interacting with the Internet of Things," *IEEE Internet Comput.*, vol. 14, no. 2, pp. 46–53, Mar. 2010.

[27] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous Internet of Things build our future: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2011–2027, 3rd Quart., 2018.

[28] K. Mori, "Automotive edge computing consortium—A global effort to develop a connected car platform," *NTT Tech. Rev.*, vol. 16, no. 6, pp. 1–4, 2018.

[29] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," 2015, *arXiv:1512.07679*.

[30] Y. Chandak, G. Theocharous, J. Kostas, S. Jordan, and P. Thomas, "Learning action representations for reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, NY, Jun. 2019, pp. 941–950.

[31] *CCPZMQ*. Accessed: Aug. 6, 2021. [Online]. Available: https://github.com/zeromq/cppzmq.git

[32] *Automotive Edge Computing Consortium General Principle and Vision White Paper*. Accessed: Aug. 6, 2021. [Online]. Available: https://aecc.org/wp-content/uploads/2019/04/AECC_White_Paper_v2.1_003.pdf

[33] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-air: When urban air quality inference meets big data," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2013, pp. 1436–1444.

[34] A. Sharma, "Intelligent querying in camera networks for efficient target tracking," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Berkeley, CA, USA, Aug. 2019, pp. 6458–6459.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[36] X. Zhang and B. Li, "Optimized multipath network coding in lossy wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 622–634, Jun. 2009.

[37] *Stable-Baseline3 in PyTorch*. Accessed: Aug. 6, 2020. [Online]. Available: https://github.com/DLR-RM/stable-baselines3.git

[38] *PyTorch*. Accessed: Aug. 6, 2020. [Online]. Available: https://github.com/pytorch/pytorch.git

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826.

[41] J. Yoon, S. Arik, and T. Pfister, "Data valuation using reinforcement learning," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, Virtual Only, IL, USA, Jul. 2020, pp. 10842–10851.

[42] *Casestudy Dataset*. Accessed: Aug. 6, 2021. [Online]. Available: https://github.com/mkris0714/Repot-Casestudy-DataSet.git

[43] P. M. de Santana, V. A. de Sousa, F. M. Abinader, and J. M. de C. Neto, "DM-CSAT: A LTE-U/Wi-Fi coexistence solution based on reinforcement learning," *Telecommun. Syst.*, vol. 71, no. 4, pp. 615–626, Aug. 2019.

[44] A. Elgabli, H. Khan, M. Krouka, and M. Bennis, "Reinforcement learning based scheduling algorithm for optimizing age of information in ultra reliable low latency networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Barcelona, Spain, Jun. 2019, pp. 1–6.

[45] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," in *Proc. IEEE Globecom*, Honolulu, HI, USA, Dec. 2019, pp. 1–6.

[46] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, Canada, Sep. 2017, pp. 23–30.

[47] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Brisbane, QLD, Australia, May 2018, pp. 3803–3810.

[48] D. P. Losey, K. Srinivasan, A. Mandlekar, A. Garg, and D. Sadigh, "Controlling assistive robots with learned latent actions," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Virtual Only, IL, USA, May 2020, pp. 378–384.

[49] D. P. Losey, H. Jun. Jeon, M. Li, K. Srinivasan, A. Mandlekar, A. Garg, J. Bohg, and D. Sadigh, "Learning latent actions to control assistive robots," 2021, *arXiv:2107.02907*.

[50] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling policies for minimizing age of information in broadcast wireless networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2637–2650, Dec. 2018.

[51] I. Kadota, A. Sinha, and E. Modiano, "Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1359–1372, Aug. 2019.

[52] R. Talak, S. KaRaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 15–28, Feb. 2019.

[53] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. 37th IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 1871–1879.

[54] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "QTCP: Adaptive congestion control with reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 445–458, Jul./Sep. 2019.

[55] K. Xiao, S. Mao, and J. K. Tugnait, "TCP-Drinc: Smart congestion control based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 11892–11904, 2019.

[56] R. Ali, N. Shahin, Y. B. Zikria, B.-S. Kim, and S. W. Kim, "Deep reinforcement learning paradigm for performance optimization of channel observation-based MAC Protocols in dense WLANs," *IEEE Access*, vol. 7, pp. 3500–3511, 2019.

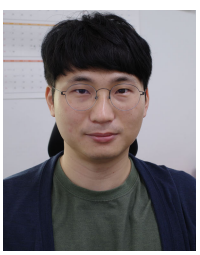
[57] Y. Guo, F. R. Yu, J. An, K. Yang, Y. He, and V. C. M. Leung, "Buffer-aware streaming in small-scale wireless networks: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6891–6902, Jul. 2019.

[58] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. 39th IEEE Conf. Comput. Commun.*, Virtual Only, IL, USA, Jul. 2020, pp. 257–266.

[59] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, and S. Levine, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proc. IEEE Int. Conf. Robot. Autom.*, Brisbane, QLD, Australia, May 2018, pp. 4243–4250.

[60] Y. Jiang, T. Zhang, D. Ho, Y. Bai, C. K. Liu, S. Levine, and J. Tan, "Sim-GAN: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Xi'an, China, May 2021, pp. 1–7.

[61] J. Truong, S. Chernova, and D. Batra, "Bi-directional domain adaptation for Sim2Real transfer of embodied navigation agents," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2634–2641, Apr. 2021.



YOUNGSEOK LEE received the B.S. degree in electric and electronic engineering from Sungkyunkwan University, Suwon, South Korea, in 2013, where he is currently pursuing the integrated Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include intelligent application, software engineering, network traffic analysis, and wireless networks. He was a recipient of the Global Ph.D. Fellowship of Korea National Research Foundation, from 2013 to 2018.

WOO KYUNG KIM received the B.S. degree from the Department of Software, Sungkyunkwan University, in 2021. He is currently pursuing the M.S. degree in computer science and engineering. His research interests include reinforcement learning, deep learning, and network system optimization.

SUNG HYUN CHOI received the B.S. degree from the Department of Software, Sungkyunkwan University, Suwon, South Korea, in 2019, where he is currently pursuing the integrated M.S. degree with the Department of Computer Science and Engineering. His research interests include intelligent application, network system optimization, and computer vision.

IKJUN YEOM received the B.S. degree in electronic engineering from Yonsei University, Seoul, South Korea, in February 1995, and the M.S. and Ph.D. degrees in computer engineering from Texas A&M University, in August 1998 and May 2001, respectively. He worked at DACOM Company, from 1995 to 1996, and Nortel Networks, in 2000. He was an Associate Professor with the Department of Computer Science, KAIST, from 2002 to 2008. Currently, he is a Full Professor with the Computer Science and Engineering Department, Sungkyunkwan University, Suwon, South Korea. His research interests include AQM, congestion control, TCP, wireless networks, and future internet architecture.

HONGUK WOO (Member, IEEE) received the B.S. degree in computer science from Korea University, Seoul, in 1995, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Austin, Austin, TX, USA, in 2002 and 2008, respectively. From 2008 to 2018, he worked at Samsung Research of Samsung Electronics as a Principal Engineer and the Vice President. Since 2018, he has been an Assistant Professor with the Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, South Korea. His research interests include intelligent application, reinforcement learning, and networked cyber-physical systems.