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

YOUNGSEOK LEE1, WOO KYUNG KIM2, SUNG HYUN CHOI2, IKJUN YEOM2, AND HONGUK WOO2

1Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, South Korea
2Department of Software, Sungkyunkwan University, Suwon, South Korea

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

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 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, Ciouonzo et al. [8] presented efficient decision fusion rules over massive multi-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 2). 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 performance.

### Table 1. A list of acronyms

| Acronym | Definition | Acronym | Definition |
|---------|------------|---------|------------|
| RepoT   | REinforcement learning POlicy with Transferability | MECH | 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 | Description | Symbol | Description |
|--------|-------------|--------|-------------|
| $D_t$  | Sensing device $i$ | $N$   | The num. of devices |
| $t$    | A discrete time-step | $I_t$ | The latest updated info. at time $t$ |
| $I_t^=$ | Aggregated infl. of all $D_i$ at $t$ | QUAL | A quality func. for monitoring quality eval. |
| $a_t$  | A bandwidth limit allocated for $D_i$ | $L_e$ | The link capacity value given environment $E$ |
| $A$    | The total BW limit | $T$  | An entire period |

**System**

- $S_t^1$: A state with all flow states for all $D_i$ at $t$  
- $a_t$: An action (a set of $a_i$) by a policy at $t$  
- $r_t^1$: A reward based on $QUAL$ with $I_t$ at $t$  
- $EMB_{\phi_0}$: Flow embedding module with parameter $\phi_0$  
- $ALLOC_{\phi_1}$: An allocation func. with parameter $\phi_1$  
- $ADJUST_{\phi_2}$: An adjustment func. with parameter $\phi_2$  
- $SHAPE$: A shaping func. with non-trainable parameter $E^\phi$: Flow embeddings by $EMB_{\phi_0}$  
- $\Delta_t^1$: A latent action by $ADJUST_{\phi_2}$  

**Implementation**

- $D$: A replay pool for RL  
- $\alpha$: A temperature parameter  
- $\pi_{\phi_0}$: A policy for $ALLOC_{\phi_1}$  
- $\pi_{\phi_2}$: A policy for $ADJUST_{\phi_2}$  
- $Q_{SAC_{in}}(s, a)$: An SAC structure for $\pi_{SAC_{in}}$ in $\in [1, 2]$  
- $Q_{SAC_{in}}$: A soft Q-func. in SAC$_{in}$ for $\pi_{SAC_{in}}$ in $\in [1, 2]$  
- $\psi_{SAC}$: A para. of flow embed. in the critic of SAC$_{in}$  
- $J_{\psi_{SAC}}(s, a)$: An objective func. for $\alpha$  
- $J_{\psi_{SAC}(s, a)}$: An actor objective func. for $\psi_{SAC}$ in SAC$_{in}$  

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI: 10.1109/ACCESS.2021.3125008, IEEE Access

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
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 I provides a list of acronyms used in this paper, and Table II 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].

![Diagram](https://example.com/diagram)

**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, and (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

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

$$I^t = (I^t_1, \ldots, I^t_{N_D}).$$

(1)

In addition, we represent the real-time monitoring quality as a function $\text{QUAL}$ 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

$$\text{maximize } \text{QoE} = \sum_{t=0}^{T-1} \text{QUAL}(I^t; a^t_1, \ldots, a^t_{N_D}),$$

subject to $\sum_{i=1}^{N_D} a^t_i = A \leq L_\mathcal{E}$

(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 \( \{a_t\}_{i=1}^{N_D} \) 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 \((S, A, p, r, \gamma)\). An MDP consists of a state space \( S \), an action space \( A \), a state transition probability \( p : S \times A \times S \rightarrow [0, 1] \), a reward \( r : S \times A \rightarrow [0, 1] \), and a discount factor \( \gamma \in [0, 1] \). We assume that \( S \) and \( A \) are continuous and \( p \) is unknown.

**State.** A state \( S \) is represented as

\[
S = \{S_1, \ldots, S_{N_D}\}
\]

(3)

where \( S_t = [s_t^0, \ldots, s_t^{N_D}]^\top \in \mathbb{R}^{d \times (u + 1)} \) denotes the flow states of individual \( D_i \) for \( d \) distinct features with a history window in \((u + 1)\) time-steps. Note that \( s_t^i \) denotes the current state of device \( D_i \), and \( s_t^{i-1} \) 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 \( u = 3 \).

**Action.** An RL-based orchestrator determines an action \( a \) using a \( \phi \)-parameterized policy \( \pi_\phi \) upon a state \( S \),

\[
a = [a_1, \ldots, a_{N_D}]^\top
\]

(4)

where \( a_i \) sets the bandwidth limit of each device \( D_i \).

**Reward.** A reward \( r \) is calculated based on the quality function \( \text{QUAL} (\cdot) \). For time-step \( t \), we have

\[
r = \text{QUAL}(I^t).
\]

(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 conditions in a given target environment. This two-phase procedure is detailed in Algorithms 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 \( a^t \) in Eq. (4) from the embeddings. In the following, we explain those modules in detail.

#### A. FLOW EMBEDDING

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

**Vectorization:** For each time-step \( t \), an intermediate embedding vector \( e^t_i \) based on a historical flow state \( S^t_i \) is obtained through \( \text{MLP}_\psi (\cdot) \), i.e.,

\[
e^t_i = \text{MLP}_\psi (S^t_i).
\]

(6)

**Relation Extraction:** Given intermediate embeddings \( E^t = [e^t_1, ..., e^t_{N_D}] \), the respective flow embeddings are obtained by

\[
E^t = \text{ATT}_\psi (E^t) = [e^t_1', ..., e^t_{N_D}'].
\]

(7)

In \( \text{ATT}_\psi (\cdot) \), query, key, and value vectors (i.e., \( q_i, k_i \), and \( v_i \)) are calculated through respective MLPs (i.e., \( \text{MLP}_\psi (\cdot) \), \( \text{MLP}_\psi (\cdot) \), and \( \text{MLP}_\psi (\cdot) \)). Given each vector in \( E^t \), its respective elements are first obtained by

\[
x_i = \text{MLP}_\psi (e^t_i)
\]

(8)

for \( x \in \{ q, k, v \} \). Then, attentive weight vector \( w \) representing the relationship between flow state \( S^t_i \) and the others is obtained using the scaled dot-production of \( q_i \) and \( [k_1, ..., k_{N_D}] \).

\[
w_i = \text{Softmax} \left( \frac{q_i^\top k_1}{\sqrt{d}}, ..., \frac{q_i^\top k_{N_D}}{\sqrt{d}} \right)
\]

(9)

where \( \text{Pr}(w_i') = \frac{\exp(w_i')}{\sum_{j} \exp(w_j')} \). Finally, flow embedding vector \( e^t_i \) is obtained by

\[
e^t_i = \sum_{j} w_{ij} v_j = \text{MLP}_\psi (S^t_i).
\]

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 note the flow embedding in a simple form combining Eq. (6) and (7).

\[
E^t = \text{EMB}_\psi (S^t_i)
\]

\[
\text{ATT}_\psi \left( \left[ \text{MLP}_\psi (S^t_i'), ..., \text{MLP}_\psi (S^t_{N_D})' \right] \right)
\]

(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.

**Allocation:** A latent action \( \tilde{a}^t = [\tilde{a}_1^t, ..., \tilde{a}_{N_p}^t] \) is obtained by

\[
\tilde{a}^t = \text{ALLOC}_{\phi_1}(E^t)
\]

(11)

where each \( \tilde{a}^t \) corresponds to weighted values on a fixed-size set of geographical points \( [p_1, ..., p_{N_p}] \) 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 \( \phi_1 \)-parameterized function trained by RL to induce policy \( \pi_{\phi_1} \).

**Adjustment:** As described in our design principle above, each latent action is transformed into a target-specific action. Specifically, control (parameter) values \( \Delta^t = [\delta_1, k_0^t, v_0^t] \) are first calculated by another policy \( \pi_{\phi_2} \) with trainable parameters \( \phi_2 \), i.e.,

\[
\Delta^t = \text{ADJUST}_{\phi_2}(E^t).
\]

(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 \( \delta \) within small \( z\% \) of that of \( \tilde{a} \), and have \( z = 30\% \) by default.
Algorithm 4 Bandwidth Allocation

1: // i. Observation for each historical flow state
2: Observe a flow state $s^i_t = [s^1_t, ..., s^N_t]$ in Eq. (3).
3: // ii. Flow embedding using the modified self-attention function
4: Compute flow embeddings $E^i = \text{EMB}_\psi(S^i)$ in Eq. (10).
5: // iii. Estimating a latent action
6: Determine a latent action $\tilde{a}^i = \text{ALLOC}_{\phi}(E^i)$ in Eq. (11).
7: // iv. Adaptation for a target network environment
8: Determine control values $[\tilde{a}_1^i, k_3^i, v_3^i]$ = \text{ADJUST}_{\phi_2}(E^i)$ in Eq. (12).
9: // v. Action shaping based on $\tilde{a}^i$ and $\Delta^t = [\tilde{a}_2^i, k_3^i, v_3^i]$
10: Compute an action $a^i = \text{SHAPE}([\tilde{a}_1^i + \tilde{a}_2^i])_{k + k_3^i, v + v_3^i}$ in Eq. (13).
11: // vi. Bandwidth allocation for each device's flow
12: Send a packet including the bandwidth limit value $a^i_t$ to device $D_i$.

Shaping: Given control values $[\tilde{a}_3^i, k_3^i, v_3^i]$ in Eq. (12), a latent action $\tilde{a}^i$ in Eq. (11) is transformed as

$$a^i = \text{SHAPE}([\tilde{a}_1^i + \tilde{a}_2^i])_{k + k_3^i, v + v_3^i}$$

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 $[a_1^i, ..., a_{N_D}^i]$ 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 $i \in \{1, ..., N_D\}$) to a set of geographical points $[p_1, ..., p_{N_P}]$, i.e.,

$$a_1^i = \sum_{j=1}^{N_P} \frac{\tilde{a}_j^i + \tilde{a}_j^i + \epsilon}{\|D_i - p_j\|^2_k + k_3^i + \epsilon},$$

$$a_2^i = \begin{cases} a_1^i, & \text{if } a_1^i \geq 0.1N_P, \\ c, & \text{otherwise}, \end{cases}$$

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 $a^i$ is obtained by transforming the intermediate values calculated above, i.e.,

$$a^i = (1 - v - v_3^i)A \cdot \text{Softmax}([a_1^i, ..., a_{N_D}^i])$$

$$= (1 - v - v_3^i)A \cdot [\text{Pr}(a_1^i), ..., \text{Pr}(a_{N_D}^i)]^T$$

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

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 $O(d \cdot N_P^2)$ 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 5 Training Rollout in Phase-1

1: // i. Establishing policy $\pi_{\phi_1}$ by SAC$_1$ in a source environment */
2: // Training parameter initialization
3: Initialize $\psi_{\text{Act}}, \psi_{\text{Critic}}, \phi_1, \alpha, \beta_{1,n}$ for $n \in \{1, 2\}$
4: $\theta_{1,n} \leftarrow \theta_{1,n}$ for $n \in \{1, 2\}$, $D \leftarrow \emptyset$
5: // Model training with total time-step $T$ and learning rate $\lambda$
6: for $t$ ← 1 to $T$
do
7: // Training sample store
8: for each environment step do
9: Sample $\tilde{a}' \sim \pi_{\phi_1}(\tilde{a}'|\text{EMB}_{\phi_2}(S^i))$.
Sample $S^{t+1} \sim \rho_{\phi_1}(S^{t+1}|S^i, \tilde{a}')$
10: Store $D \leftarrow \emptyset \cup \{(S^t, \tilde{a}', r^t, S^t+1)\}$
11: end for
12: // Parameter update
13: for each gradient step do
14: // Soft Q-function and critic’s embedding module update
15: Update $\theta_{1,T} \leftarrow \theta_{1,T} - \nabla_{\theta_{1,T}} J_{\text{Critic}}^T(S^t, \tilde{a}', \lambda)$
16: // Policy and actor’s embedding module update
17: Update $\phi_1 \leftarrow \phi_1 - \nabla_{\phi_1} J_{\text{Actor}}^T(\lambda)$.
$\psi_{\text{Actor}} \leftarrow \psi_{\text{Actor}} - \nabla_{\psi_{\text{Actor}}} J_{\text{Actor}}^T(\lambda)$
18: // Temperature parameter update
19: Update $\alpha \leftarrow \alpha - \hat{\alpha} \cdot J_{\text{Actor}}^T(\lambda)$
20: // Target soft Q-function update
21: Update $\theta_{1,T} \leftarrow \chi \theta_{1,T} + (1 - \chi)\theta_{t-1}$ for $n \in \{1, 2\}$
22: end for
23: end for

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 $\psi_{\text{Act}}$ and $\phi_1$, and the critic includes the model parameters $\psi_{\text{Critic}}$ and $\theta_1$. SAC$_1$ drives all those parameters to be updated in an end-to-end training fashion. Note that $\theta_1$ 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 $\psi_{\text{Act}}$ and $\phi_2$ and the critic includes the model parameters $\psi_{\text{Critic}}$ and $\theta_2$, SAC$_2$ drives only the parameters $\phi_2$ and $\theta_2$ to be updated during its training; the other model parameters are fixed. Same as the two soft Q-functions of $\theta_1$, we have $\theta_{2,n}$ for $n \in \{1, 2\}$. The
Algorithm 6 Training Rollout in Phase-2

1: // Training policy πφm,n by SAC-2 in a target environment 
2: // Training parameter load and initialization 
3: Load ψCrx, ψAct, φ1, Initialize φ2, φ2,n for n ∈ {1, 2} 
4: \( \bar{\theta}_{2,n} \leftarrow \theta_{2,n} \) for n ∈ {1, 2}, \( D \leftarrow \emptyset \)
5: // Fine-tuning with total time-step \( T' \) and learning rate \( \lambda' \) 
6: for \( t \leftarrow 1 \) to \( T' \) do 
7: // Training sample store 
8: for each environment step do 
9: Sample \( \Delta' \sim \pi_{\sigma} \Delta' \) | EMB \( \psi_{\text{Act}}(S') \)| \( \pi_{\sigma} \) 
10: Sample \( S^{t+1} \sim \rho_{\sigma} \Delta' \) | \( \pi_{\sigma} \) 
11: Store \( D \leftarrow D \cup \{(S^t, \Delta^t, r^t, S^{t+1})\} \)
12: end for 
13: // Parameter update 
14: for each gradient step do 
15: // Soft Q-function update 
16: Update \( \theta_{2,n} \leftarrow \theta_{2,n} - \lambda' \nabla \psi_{\sigma} \cdot \pi_{\sigma} \pi_{\sigma} \) for n ∈ {1, 2} 
17: // Adaptation policy update 
18: Update \( \phi_2 \leftarrow \phi_2 - \lambda' \nabla \psi_{\sigma} \pi_{\sigma} \phi_{\sigma} \) 
19: // Temperature parameter update 
20: Update \( \alpha \leftarrow \alpha - \lambda' \nabla \psi_{\sigma} \pi_{\sigma} \phi_{\sigma} \) 
21: // Target soft Q-function update 
22: Update \( \bar{\theta}_{2,n} \leftarrow \chi \bar{\theta}_{2,n} + (1 - \chi) \bar{\theta}_{2,n} \) for n ∈ {1, 2} 
23: end for 
24: end for

declares the learning target of SAC-2 in a target environment. The algorithm iterates through the environment's states and actions, updating the parameters \( \theta_{2,n} \) and \( \phi_2 \) to improve the policy and Q-function. The training process involves sampling trajectories and updating the policy and Q-function parameters to minimize the Bellman residual.

In evaluation, we consider a networked monitoring system where status updates of sensing devices are aggregated for continual query processing. We build a simulation environment for such network monitoring based on NS-3 [25] v3.29, a widely used packet-level discrete-event network simulator, and 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(\cdot) in Eq. 1, we use structural similarity index measure (SSIM) [35] that estimates the similarity between the aggregated status information \( I' \) (at the server) and the

**Algorithm 6 Training Rollout in Phase-2**

1. // Training policy πφm,n by SAC-2 in a target environment
2. // Training parameter load and initialization
3. Load ψCrx, ψAct, φ1, Initialize φ2, φ2,n for n ∈ {1, 2}
4. \( \bar{\theta}_{2,n} \leftarrow \theta_{2,n} \) for n ∈ {1, 2}, \( D \leftarrow \emptyset \)
5. // Fine-tuning with total time-step \( T' \) and learning rate \( \lambda' \)
6. for \( t \leftarrow 1 \) to \( T' \) do
7. // Training sample store
8. for each environment step do
9. Sample \( \Delta' \sim \pi_{\sigma} \Delta' \) | EMB \( \psi_{\text{Act}}(S') \)| \( \pi_{\sigma} \)
10. Sample \( S^{t+1} \sim \rho_{\sigma} \Delta' \) | \( \pi_{\sigma} \)
11. Store \( D \leftarrow D \cup \{(S^t, \Delta^t, r^t, S^{t+1})\} \)
12. end for
13. // Parameter update
14. for each gradient step do
15. // Soft Q-function update
16. Update \( \theta_{2,n} \leftarrow \theta_{2,n} - \lambda' \nabla \psi_{\sigma} \cdot \pi_{\sigma} \pi_{\sigma} \) for n ∈ {1, 2}
17. // Adaptation policy update
18. Update \( \phi_2 \leftarrow \phi_2 - \lambda' \nabla \psi_{\sigma} \pi_{\sigma} \phi_{\sigma} \)
19. // Temperature parameter update
20. Update \( \alpha \leftarrow \alpha - \lambda' \nabla \psi_{\sigma} \pi_{\sigma} \phi_{\sigma} \)
21. // Target soft Q-function update
22. Update \( \bar{\theta}_{2,n} \leftarrow \chi \bar{\theta}_{2,n} + (1 - \chi) \bar{\theta}_{2,n} \) for n ∈ {1, 2}
23. end for
24. end for

 denotes the parameter of a target soft Q-function, obtained as an exponentially moving average of the soft Q-function weights.

**Actor:** We optimize policy \( \pi_{\phi_m} \) and the flow embedding module \( EMB_{\psi_{\text{Act}}} \) in Eq. 10 with the parameters \( \psi_{\text{Act}} \) by minimizing the below objective function,

\[
J_{\text{Act}}(m) = \mathbb{E}_{S^t \sim D} \left[ \alpha \log \frac{\pi_{\phi_m}(f_{\phi_m}(e^t; EMB_{\psi_{\text{Act}}}(S^t)) | EMB_{\psi_{\text{Act}}}(S^t))}{Q_{\text{Min}}(EMB_{\psi_{\text{Crx}}}(S^t), f_{\phi_m}(e^t; EMB_{\psi_{\text{Act}}}(S^t)))} \right] 
\]  

(19)

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

**Temperature parameter:** To improve the performance and stability of the SAC algorithm, we use the following objective to calculate gradients for temperature parameter \( \alpha \),

\[
J_{\alpha}(m) = \mathbb{E}_{\tilde{a}^t \sim \pi_{\phi_m}} \left[ -\alpha \log \frac{\pi_{\phi_m}(\tilde{a}^t | EMB_{\psi_{\text{Act}}}(S^t))}{\alpha \tilde{H}} \right] 
\]  

(20)

where \( \tilde{H} \) 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(\cdot) in Eq. 1, we use structural similarity index measure (SSIM) [35] that estimates the similarity between the aggregated status information \( I' \) (at the server) and the
TABLE 3. Experiment network conditions for source environment $B$ and target environment $T$. The default settings for $T$ are in parentheses.

| Config.                        | Wireless network env. | Source env. $B$ | Target env. $T$ |
|--------------------------------|-----------------------|-----------------|-----------------|
| 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      |                 |

Note that ground truth $I_{\text{truth}}^t$ (at the sensing devices) at time $\tau$ within a time-step interval.

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

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

(21)

Note that $\mu$ and $\sigma$ are the mean and standard deviation of $x = I^t$ and $y = I_{\text{truth}}^t$, 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 4 illustrates the environment settings where the source $B$ denotes a learning environment and the target $T$ 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 $B$ is set to have easy-to-learn network conditions characterized as ideal time-slotted communications 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 $T$ 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 Nis- tErrorRateModel, ConstantSpeedDelayModel, and LogDistanceLossModel.

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-baseline [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 $a_i$. This method is used to set the reference performance for comparison.
- **Random.** Each device is assigned a randomly distributed $a_i$ 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.,

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

(22)

where $\text{OB}()$ 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-function takes those features as input to determine the next action.
TABLE 5. Hyperparameter settings for RL-based methods

| Hyperparameter          | Value         |
|-------------------------|---------------|
| Optimizer               | Adam          |
| Discount factor $\gamma$| 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 $\lambda$ | 0.00005      |
| Adaptation learning rate $\lambda'$ | 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       |

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.

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 $M$, the relative QoE is calculated as

$$\text{Relative QoE} (\%) = \frac{\text{QoE}_M}{\text{QoE}_{\text{Uniform}}} \times 100 \quad (23)$$

where $\text{QoE}_{\text{Uniform}}$ denotes the ideal reference QoE and $\text{QoE}_M$ denotes the achieved QoE by the method $M$. 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 $B$ and target $T$, this reference network is set to have neither channel access delay nor propagation delay.

Figure 3 represents the performance of Repot and the other methods which are trained and tested in $B$ 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 [29] 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.

In Figure 4, we illustrate the 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.

VOLUME 9, 2021
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 $T$, we train the models of Naïve-RL, ARN-RL, and Repot in the source environment $B$ with the dynamic link capacity of $\frac{1}{2} L_B, L_B$, where $L_B$ is the link capacity of $B$. Then, we fine-tune each learned model (policy) $M$ in $T$ 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 ADJUST$_{top}(\cdot)$ in Eq. 12 is updated. In addition, we update the $K$ value of Top-$K$ to be optimized in $T$ for Top-Opt.

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

This performance achieved in $T$ by Repot is not only better than that of the other methods but also comparable to that in $B$ by itself (i.e., 91.0% in $M_{Base}(B)$ and 90.7% in $M_{FT}(T)$); 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 $T$ 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 Repot$_{FT}$ 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 ARN-RL$_{FT}$ and Repot$_{FT}$ during the first period of learning steps due to our learning with domain randomization. Furthermore, the limited performance improvement of Naïve-RL$_{FT}$ and ARN-RL$_{FT}$ 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 $T^*_I$ 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 $Repot_{\text{Base}}$, we use the base model optimized in $B$ without fine-tuning. In $Repot_{\text{AS}}$, we use our proposed action shaping, while we use other fine-tuning schemes in $Repot_{\text{Top}}$ and $Repot_{\text{full}}$. Specifically, $Repot_{\text{Top}}$ updates the last layer parameters in the ALLOC function, similar to conventional transfer learning, and $Repot_{\text{full}}$ updates its full layer parameters for fine-tuning, considering significant differences between source and target environments.

Overall, $Repot_{\text{AS}}$ achieves robust performance across all $T^*_I$ with $T^*_4$, showing highly comparable adaptation performance in $T^*_I$ to the respective one in $B$. The other fine-tuned models $Repot_{\text{Top}}$ and $Repot_{\text{full}}$ 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 $Repot_{\text{AS}}$ are much different from those of $Repot_{\text{Base}}$ than the others, and they often tend to spread. We speculate that $Repot_{\text{AS}}$ has a more significant policy to target environments that involve uncertainty more than our source environment due to realistic network settings.

### Table 6. Dataset characteristics

| Config. | Urban-scale | Suburb-scale |
|---------|-------------|--------------|
| Map size | 10.1 km² | 5.9 km² |
| 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 |

### 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 on the maps of Midtown Manhattan in New York and a suburb of Orlando, Florida. We make the datasets publicly available on Github. Table 5 describes the configuration to generate the datasets, where the urban-scale datasets have about 10 times heavier traffic than the suburb-scale datasets. 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 8](image-url) **FIGURE 8.** Learning graph of different fine-tuning schemes: The learning graphs correspond to the fine-tuning process for the target environment $T^*_I$ by different Repot models (with different fine-tuning approaches) in Figure 7.

![Figure 9](image-url) **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.

![Figure 10](image-url) **FIGURE 10.** Performance comparison in real traffic scenarios: The case study, urban-scale in (a) and suburb-scale in (b). We conduct fine-tuning the $Repot$ model achieved in the source $B$ for the target $T$ using 6.25% of the training sample amount used in $B$. The $Repot$ model yields the best performance (i.e., 96.1% in (a) and 99.6% in (b)) in $T$. More importantly, the fine-tuned $Repot$ model yields the best performance (i.e., 94.9% in (a) and 93.9% in (b)) in $T$. Whereas the $Repot$ model before fine-tuning in $T$ 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 $\text{Top-Opt}$ (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.
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. [50] | 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 |

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 Communications Magazine, 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 Communications, vol. 27, no. 1, pp. 108–115, 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 of Things Journal, vol. 7, no. 9, pp. 8748–8759, 2020.
[4] S. Kaul, R. Yates, and M. Gruteser, “Real-time status: How often should one update?” in Proc. of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM), Orlando, FL, 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. of the 38th IEEE International Conference on Computer Communications (INFOCOM), 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 Transactions on Wireless Communications, vol. 11, no. 5, pp. 1744–1752, 2012.
[7] D. Cionzio, G. Campano, and P. S. Rossi, “Channel-aware decision fusion in distributed MIMO wireless sensor networks: Decode-and-fuse vs. decode-then-fuse,” IEEE Transactions on Wireless Communications, vol. 11, no. 8, pp. 2976–2985, 2012.
[8] D. Cionzio, P. S. Rossi, and S. Dey, “Massive MIMO channel-aware decision fusion,” IEEE Transactions on Signal Processing, vol. 63, no. 3, pp. 604–619, 2015.
[9] A. Deshpande, C. Guerin, S. R. Madden, J. M. Hellerstein, and W. Hong, “Model-driven data acquisition in sensor networks,” in Proc. of the 30th International Conference on Very Large Data Bases (VLDB ’04), Tokyo, Japan, Aug. 2004, pp. 588–599.
[10] J. Traub, S. Brell, T. Rabl, A. Katisfodimos, and V. Markl, “Optimized on-demand data streaming from sensor nodes,” in Proc. of the 2017 ACM Symposium on Cloud Computing (SoCC ’17), Santa Clara, California, 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 of Things Journal, vol. 8, no. 14, pp. 11 329–11 344, 2021.
[12] P. Whittle, “Restless bandits: Activity allocation in a changing world,” Journal of Applied Probability, vol. 25, no. A, pp. 287–298, 1988.
[13] S. M. Kay, “Fundamentals of statistical signal processing: Estimation theory,” USA: Prentice-Hall PTR, 1993.
[14] ———, “Fundamentals of statistical signal processing: Detection theory,” USA: Prentice-Hall PTR, 1998.
[15] A. Beloglazov and R. Buyya, “Energy efficient resource management in virtualized cloud data centers,” in Proc. of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid ’10), Melbourne, 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,” Mathematical and Computer Modelling, vol. 58, no. 5–6, pp. 1222–1235, Sep. 2013.
[17] H. Mao, M. Alizadeh, I. Menache, and S. Kendula, “Resource management with deep reinforcement learning,” in Proc. of the 15th ACM Workshop on Hot Topics in Networks (HotNets ’16), Atlanta, GA, 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. of the ACM Special Interest Group on Data Communication (SIGCOMM ’19), 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, “Oflow: A reinforcement learning approach to high QoE video streaming over wireless networks,” in Proc. of the 20th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc’19), Catania, Italy, July 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. 7442–74441, 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. of the 38th IEEE International Conference on Computer Communications (INFOCOM), 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,” CoRR, vol. abs/1812.05905, 2018. [Online]. Available: http://arxiv.org/abs/1812.05905.
[24] D. Krajzewicz, G. Hertkorn, P. Wagner, and C. Rüssel, “SUMO (simulation of urban mobility): An open-source traffic simulation,” in Proc. of the 4th Middle East Symposium on Simulation and Modelling, Berlin, Germany, Sep. 2002, pp. 183–187.
[25] “NS-3.” [Online]. Available: https://github.com/nsnam/ns-3-dev.git. Accessed on: Aug. 6, 2021.
[26] M. Kranz, P. Holleis, and A. Schmidt, “Embedded interaction: Interacting with the internet of things,” IEEE Internet Computing, vol. 14, no. 2, pp. 46–53, 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 Communications Surveys Tutorials, vol. 20, no. 3, pp. 2011–2027, 2018.
[28] K. Mori, “Automotive edge computing consortium—a global effort to develop a connected car platform,” NTT Technical Review, vol. 16, no. 6, pp. 1–4, 2018.
[29] G. Dulac-Arnold, R. Evans, P. Sunehag, and B. Coppin, “Reinforcement learning in large discrete action spaces,” CoRR, vol. abs/1512.07679, 2015. [Online]. Available: http://arxiv.org/abs/1512.07679.
[30] Y. Chandak, G. Theocharous, J. Kostas, S. Jordan, and P. Thomas, “Learning action representations for reinforcement learning,” in Proc. of the 36th International Conference on Machine Learning (ICML), Long Beach, NY, June 2019, pp. 941–950.
[31] “CPPZMQ,” [Online]. Available: https://github.com/zeromq/cppzmq.git. Accessed on: Aug. 6, 2021.
[32] “Automotive edge computing consortium general principle and vision white paper,” [Online]. Available: https://aecc.org/wp-content/uploads/2019/04/AECC_White_paper_v2.1_003.pdf. Accessed on: Aug. 6, 2021.
[33] Y. Zheng, F. Liu, and H.-P. Hsieh, “U-Air: When urban air quality meets big data,” in Proc. of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’13), New York, NY, Aug. 2013, pp. 1436–1444.
[34] A. Sharma, S. Anand, and S. Kaul, “Reinforcement learning based querying in camera networks for efficient target tracking,” in Proc. of the International Conference on Automated Planning and Scheduling, Berkeley, CA, June 2019.
[35] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, 2004.
[36] X. Zhang and B. Li, “Optimized multipath network coding in lossy wireless networks,” IEEE Journal on Selected Areas in Communications, vol. 27, no. 5, pp. 622–634, 2009.

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

[38] “PyTorch,” [Online]. Available: https://github.com/pytorch/pytorch.git.

[39] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” Cambridge, MA: MIT Press, 1998.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proc. the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, June 2016, pp. 2818–2826.

[41] J. Yoon, S. Arik, and T. Pfister, “Data valuation using reinforcement learning,” in Proc. the 37th International Conference on Machine Learning (ICML), Virtual Only, July 2020, pp. 10842–10851.

[42] “Casestudy dataset,” [Online]. Available: https://github.com/mkris0714/Repot-Casestudy-DataSet.git. Accessed on: Aug. 6, 2021.

[43] P. M. de Santana, V. A. de Sousa Jr., F. M. A. Jr., 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, 2019.

[44] A. Elgabli, H. Khan, M. Krouka, and M. Bennig, “Reinforcement learning based scheduling algorithm for optimizing age of information in ultra reliable low latency networks,” in Proc. of the 24th IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, June 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. of the 2019 IEEE Global Communications Conference (GLOBECOM), Honolulu, HI, 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. of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, 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. of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, May 2018, pp. 3803–3810.

[48] D. P. Losey, K. Srivivasan, A. Mandlekar, A. Garg, and D. Sadigh, “Controlling assistive robots with learned latent actions,” in Proc. of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Virtual Only, May 2020, pp. 378–384.

[49] D. P. Losey, H. J. Jeon, M. Li, K. Srivivasan, A. Mandlekar, A. Garg, J. Bohg, and D. Sadigh, “Learning latent actions to control assistive robots,” CoRR, vol. abs/2107.02907, 2021. [Online]. Available: https://arxiv.org/abs/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 Transactions on Networking, vol. 26, no. 6, pp. 2637–2650, 2018.

[51] I. Kadota, A. Sinha, and E. Modiano, “Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints,” IEEE/ACM Transactions on Networking, vol. 27, no. 4, pp. 1359–1372, 2019.

[52] R. Talak, S. Karaman, and E. Modiano, “Optimizing information freshness in wireless networks under general interference constraints,” IEEE/ACM Transactions on Networking, vol. 28, no. 1, pp. 15–28, 2020.

[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. of the 37th IEEE Conference on Computer Communications (INFOCOM 2018), Honolulu, HI, 4 2018, pp. 1871–1879.

[54] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, “QTCP: Adaptive congestion control with reinforcement learning,” IEEE Transactions on Network Science and Engineering, vol. 6, no. 3, pp. 445–458, 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. 11 892–11 904, 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 Transactions on Vehicular Technology, vol. 68, no. 7, pp. 6981–6992, 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. of the 39th IEEE Conference on Computer Communications (INFOCOM 2020), Virtual Only, July 2020, pp. 257–266.

[59] K. Bousmalis, A. Ipan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, “Using simulation and domain adaptation to improve efficiency of deep robotic grasping,” in Proc. of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, May 2018, pp. 4243–4250.

[60] Y. Jiang, T. Zhang, D. Ho, Y. Bai, C. K. Liu, S. Levine, and J. Tan, “Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning,” in Proc. of the 2021 IEEE International Conference on Robotics and Automation (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 Robotics and Automation Letters, vol. 6, no. 2, pp. 2634–2641, 2021.