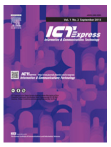
The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 10 (2024) 576–582

www.elsevier.com/locate/icte



Efficient deep reinforcement learning under task variations via knowledge transfer for drone control

Sooyoung Jang, Hyung-Il Kim[*]

Department of Computer Engineering, Hanbat National University, Daejeon, South Korea
Visual Intelligence Research Section, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea

Received 24 August 2023; received in revised form 10 March 2024; accepted 8 April 2024
Available online 10 April 2024

Abstract

Despite the growing interest in using deep reinforcement learning (DRL) for drone control, several challenges remain to be addressed, including issues with generalization across task variations and agent training (which requires significant computational power and time). When the agent's input changes owing to the drone's sensors or mission variations, significant retraining overhead is required to handle the changes in the input data pattern and the neural network architecture to accommodate the input data. These difficulties severely limit their applicability in dynamic real-world environments. In this paper, we propose an efficient DRL method that leverages the knowledge of the source agent to accelerate the training of the target agent under task variations. The proposed method consists of three phases: collecting training data for the target agent using the source agent, supervised pre-training of the target agent, and DRL-based fine-tuning. Experimental validation demonstrated a remarkable reduction in the training time (up to 94.29%), suggesting a potential avenue for the successful and efficient application of DRL in drone control.

Keywords: Deep reinforcement learning; Drone control; Task variations; Knowledge transfer

1. Introduction

Recently, the development of deep reinforcement learning (DRL) algorithms has enabled drones to excel in various applications, such as navigation [1,2], obstacle avoidance [3], drone racing [4], drone taxi [5], and patrolling [6]. In these DRL-based approaches, drones act as autonomous agents. They interact with their environment through trial and error and are optimized by feedback in the form of rewards. However, a recurring challenge is the need for DRL-based drone agents to adapt seamlessly to various tasks, as the sensors or missions in real-world applications often vary. Predictably, a drone agent optimized for RGB sensor inputs will suffer a significant performance drop when the sensor is switched to depth or RGB-D due to its lack of knowledge of these new input patterns. Similarly, a drone agent optimized for reaching a target given its position (target navigation) may perform poorly if its mission changes to reaching the target using only image sensors without knowing its position (target search). The core issue is the limited ability of a DRL-based drone agent to adapt across different tasks, frequently necessitating resource-intensive retraining from scratch.

Potential solutions, such as knowledge distillation [7–9], domain-adaptive DRL [10], imitation learning [11,12], and meta-learning [13], have been explored. [8] introduced KnowRU, a method for efficient knowledge reuse in multi-agent reinforcement learning, leveraging knowledge distillation [7]. It utilizes the tuples of (s_i, a_i, r_i) from the teacher's replay buffer to train the student's actor and critic networks. [9] explored various distillation methods and proposed expected entropy regularized distillation, which incorporates an entropy term in the loss function to accelerate learning. In [10], the agents were trained to generate disentangled representations invariant to domain shifts. In particular, they aim to ensure that, once trained in a source domain, an agent can generalize its learning to a new, unencountered target domain with minimal to no additional training. In [11], an imitation learning algorithm was proposed to disentangle visual mappings from game dynamics, thereby enhancing the

[*] Corresponding author at: Visual Intelligence Research Section, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea.

E-mail addresses: syjang@hanbat.ac.kr (S. Jang), hikim@etri.re.kr (H.-I. Kim).

Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2024.04.002

visual transfer-based policies. Similarly, BAIL [12] leverages imitation learning to select actions that are likely to yield high performance, and these selected actions are subsequently employed to train a policy network. [13] proposed a meta-learning-based domain-adaptation technique for adapting agents across different domains with minimal trials. Although significant research progress has been achieved, a crucial gap remains in addressing the unique challenges posed by task variations in DRL. This gap is particularly pronounced in drone control, where missions (target navigation and target search) and sensors (depth, RGB to RGB-D) vary frequently, altering the observation and action spaces, and consequently, the neural network architecture.

To tackle the challenge of limited adaptability in DRL-based drone agents, we introduce an efficient DRL method that enhances training speed and flexibility through knowledge transfer. This method involves transferring knowledge from a drone agent trained on one task (the source agent) to another that needs to learn a different task (the target agent). Our approach is structured into three phases: initially, we collect data with the source agent, which is then used to pre-train the target agent under supervision; following this, we fine-tune the target agent using DRL. This methodology was evaluated across four drone control tasks, each with distinct mission and sensor setups, and we observed a substantial reduction in training times. Our contributions are as follows:

- We introduce a novel knowledge transfer method that significantly speeds up the training process. By using data from a source agent for pre-training, followed by DRL-based fine-tuning, we achieved up to a 94.29% reduction in training time.
- Our method offers flexibility in the neural network architecture, including input layers between the source and target agents. This means that the architectures of the source and target agents do not need to be identical, allowing for adjustments and customization to meet the specific needs of each mission or to accommodate technical requirements.

The remainder of this paper is structured as follows: We detail our proposed method in Section 2, followed by a description of our experimental setups and results in Section 3. Finally, concluding remarks are presented in Section 4.

2. Efficient DRL under task variations

In this section, we describe an efficient DRL method for task variation. The task variations covered in this study include variations in the mission of the drone and sensors utilized in the drone. These variations inherently cause changes in the observation space between the source and target agents (e.g., a 194-dimensional vector to 84×84×3 images), which can lead to changes in the neural network architecture (e.g., a fully connected network to convolutional neural networks). The proposed method is applicable regardless of these changes. The proposed method consists of three phases: (1) data collection, (2) supervised pre-training, and (3) DRL-based fine-tuning. It focuses on transferring knowledge from a trained source agent to a randomly initialized target agent to accelerate the training of the target agent for the target task.

2.1. Phase 1: Data collection

We assume that the trained policy (π_ψ) and value (V_φ) networks of the source agent are provided and that the observation spaces of the source task ($\mathbb{O}^S$) and target task ($\mathbb{O}^T$) are included in the state space ($\mathbb{S}$) of the environment, i.e., $\mathbb{O}^S \in \mathbb{S}$ and $\mathbb{O}^T \in \mathbb{S}$.

In Phase 1, we collect data by rollout with π_ψ in the environment. The main idea in this phase is that the source agent determines the action and rollout based on observations of the source task. However, we collected observations for the target task instead of the source task. Specifically, for each rollout step t, the tuple in the format (o_t^T, y_t, v_t) is collected, where o_t^T is the observation of the target task while (y_t) and (v_t) are the action logit and value determined by the observation of the source task (o_t^S):

$$y_t = \pi_\psi(o_t^S),\ \ v_t = V_\varphi(o_t^S). \tag{1}$$

This tuple may be mismatched because o_T is not used to calculate y_t and v_t. This process has several significant advantages.

- We can easily and quickly obtain the target task's data with the source task's trained agent.
- We can freely configure the target agent (e.g., neural network architecture) as the target agent is not involved in the process.
- The pre-trained agent itself performs well, as detailed in the experiments section.

This phase continues until the termination condition for data collection, $cond_{collect}$, is satisfied. The possible conditions could be a 100-episode rollout, which we utilized for the experiments, data size, or time.

2.2. Phase 2: Supervised pre-training

We begin by setting the training data and leveraging the tuples acquired in Phase 1. The action logit and value of the source observation are used as the labels of the target observation for the supervised pre-training of the target agent's policy and value neural network, respectively. The target agent's policy and value networks are randomly initialized with parameters ψ and φ, respectively. Subsequently, we set the loss functions $\mathcal{L}_\pi$ and $\mathcal{L}_v$ to guide the iterative updating of these parameters. The loss functions for pre-training are defined as follows:

$$\mathcal{L}_\pi = \frac{1}{|O^T|} \sum_{o_t^T \in O^T} \left\| \pi_\psi(o_t^T) - y_t \right\|_2^2, \tag{2}$$

$$\mathcal{L}_v = \frac{1}{|O^T|} \sum_{o_t^T \in O^T} \left(V_\varphi(o_t^T) - v_t \right)^2, \tag{3}$$

where $|\cdot|$ denote the cardinality of the set. With a learning rate α, the gradient descent method updates the parameters to reduce the loss. Pre-training continues until the predefined condition, $cond_{pre-train}$, such as an early stopping with the patience of 30 utilized for the experiments, is satisfied.

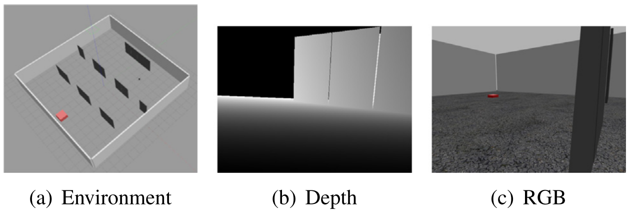


Fig. 1. Environment and a sample of the depth and RGB images captured from the sensors in the drone. They are used for training the drone agent via DRL.

2.3. Phase 3: DRL-based fine-tuning

Using DRL with an agent pre-trained from the source, we can reduce the computation time by enabling effective exploration based on the transferred knowledge. In this phase, we fine-tuned the pre-trained target agent from Phase 2 via DRL. We used proximal policy optimization (PPO) [14] as the DRL algorithm. Specifically, the agent performs a rollout to collect experiences stored as tuples of observations, actions, rewards, and subsequent observations, denoted by $\{(o_t^T, a_t, r_t, o_{t+1}^T)\}$, where a_t denotes the action sampled from Softmax($\pi_{\hat{\psi}_n}(o_t^T)$). Subsequently, the parameters of the policy $\pi_{\hat{\psi}}$ and value $V_{\hat{\varphi}}$ networks are updated by maximizing the policy objective and minimizing the value loss. Refer to [14] for a description of policy objectives and value loss. This phase persists until the termination condition, $cond_{fine-tune}$, is satisfied.

In summary, the proposed method, structured as Algorithm 1, presents a systematic approach for transferring knowledge from the source to the target tasks, providing efficient training under task variations. Moreover, the rollout processes in phases 1 and 3 can be accelerated by increasing the number of rollout workers. Six rollout workers were used in this experiment. Additional details of the experimental settings used to validate our approach, including the missions, neural network architectures, and input descriptions, are described in the following section.

3. Experiments

3.1. Experimental settings

In this study, we investigated two drone-control scenarios: target navigation and target search. Although the primary mission in both scenarios is to reach the target object while avoiding obstacles, the operational methods are different. The environment and samples of the depth and RGB images are shown in Fig. 1. A red box in Fig. 1(a) illustrates the target object.

- **Target Navigation (TN)**
  Mission Considering the target's position, the drone uses the position and sensor input to navigate to the target.
  Network Architecture For this scenario, we implemented a fully connected network (FCN) with two hidden layers, each with 256 nodes.

Algorithm 1: Pseudo-code of the proposed method.

```
Input: The trained policy ($\pi_\psi$) and value ($V_\varphi$) networks
  for the source task
Output: The trained policy ($\pi_{\hat{\psi}}$) and value ($V_{\hat{\varphi}}$)
  networks for a target task
# Phase 1: Data Collection
for $e = 1, 2, \cdots$ do
    Initialize the state $s_0$;
    for $t = 0, 1, 2, \cdots$ do
        Retrieve the observations for the source task $o_t^S$
          and the target task $o_t^T$ from state $s_t$;
        Compute the action logit $y_t = \pi_\psi(o_t^S)$ and the
          value $v_t = V_\varphi(o_t^S)$;
        Store tuple $(o_t^T, y_t, v_t)$;
        Sample action $a_t \sim Softmax(y_t)$;
        Forward $(t+1)$th step with $a_t$;
        Retrieve the next state $s_{t+1}$;
        if $done$ then
            break;
        end
    end
    if $cond_{collect}$ then
        break;
    end
end
# Phase 2: Supervised Pre-training
Set the training data for the policy network as $(o_t^T, y_t)$;
Set the training data for the value network as $(o_t^T, v_t)$;
Initialization: $\psi_0' \leftarrow \psi$, $\varphi_0' \leftarrow \varphi$;
for $n = 0, 1, 2, \cdots$ do
    Compute the loss $\mathcal{L}_\pi$ in Eq. (2);
    Compute the loss $\mathcal{L}_v$ in Eq. (3);
    (Update $\psi'$): $\psi_{n+1}' \leftarrow \psi_n' - \alpha\nabla_{\psi'}\mathcal{L}_\pi$;
    (Update $\varphi'$): $\varphi_{n+1}' \leftarrow \varphi_n' - \alpha\nabla_{\varphi'}\mathcal{L}_v$;
    if $cond_{pre-train}$ then
        break;
    end
end
# Phase 3: DRL-based Fine-tuning
Initialization: $\hat{\psi}_0 \leftarrow \psi'$, $\hat{\varphi}_0 \leftarrow \varphi'$;
for $n = 0, 1, 2, \cdots$ do
    Rollout with $\pi_{\hat{\psi}_n}$ for $T$;
    We store the tuple $(o_t^T, a_t, r_t, o_{t+1}^T)$ Update policy $\hat{\psi}_n$
      by maximizing the policy objective;
    Update the value $\hat{\varphi}_n$ by minimizing value loss;
    if $cond_{fine-tune}$ then
        break;
    end
    $\hat{\psi}_{n+1} \leftarrow \hat{\psi}_n$, $\hat{\varphi}_{n+1} \leftarrow \hat{\varphi}_n$;
end
```

Input Description The network processes a 194-dimensional input vector. In particular, 192 dimensions are derived from a 640×480 depth image downscaled by a factor of 1/40 along both axes. The remaining two dimensions quantify the distance and angular direction relative to the target.

- **Target Search (TS)**
  Mission Without prior knowledge of the target's position, the drone must search the environment for the target, relying solely on its sensor inputs. This scenario is more challenging than the target navigation scenario.
  Network Architecture Considering the visual input, we implemented a convolutional neural network (CNN) with three hidden layers. Refer to the experimental setup in [15] for details.
  Input Description This CNN assumes a resolution of 84×84 pixels, downscaled from an original 640×480 image. The channel configuration is adapted to the type of sensor used. In our experiments, we tested with 1-channel depth, 4-channel RGB-D, and 3-channel RGB sensors.

We used target navigation as the source task and target search as the target task to emphasize the efficacy of the proposed method. We showed that we could significantly reduce the training time for (more difficult) target search by using the knowledge of (easier) target navigation. Please refer to the experimental results presented in the following subsections. We employed the PPO algorithm [14] as the DRL algorithm to train the agents from scratch (baseline) and fine-tuning the pre-trained target agents (proposed). The training termination criterion was a success rate of 90% or higher for five consecutive training iterations.

3.2. Hardware and software specifications

The experiments were conducted on a Dell Precision 7920 equipped with an Intel Xeon Gold 6240 processor, 64 GB of RAM, and an Nvidia Quadro RTX 8000 with 48 GB of graphics memory. The operating system installed on the workstation is Ubuntu 18.04. Gazebo 9 and ROS Melodic were used for simulation purposes. The algorithm was developed using Ray [16] (version 1.13), which incorporates RLlib [17] with a PPO implementation, and TensorFlow (version 2.8) was used for execution. Leveraging Ray, we employed six simulators (workers) running in parallel to collect data and expedite the DRL process across all experiments that required simulators, namely the Baseline and Phases 1 and 3 of the proposed method.

3.3. Experiment I: Baseline (train from scratch)

This subsection outlines the baseline results of training a drone agent using DRL. For the baseline DRL experiment, we initialized the agent training from scratch using a randomly initialized policy and a value network across four distinct tasks. These tasks consisted of one target navigation task and three target search tasks employing depth, RGB-D, and RGB sensors. These baseline results provided a benchmark for evaluating the efficacy of the proposed methodology, which is detailed in the following subsections.

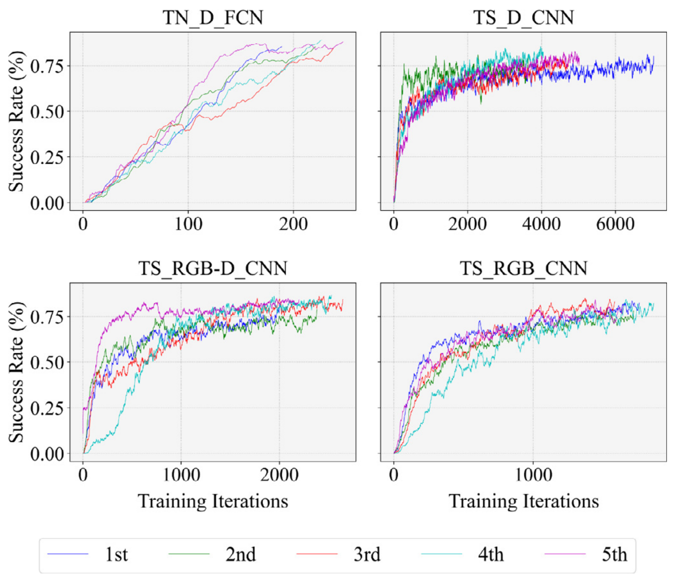


Fig. 2. Success rate (%) over training iteration for baseline. Smoothed using an exponential weighted moving average. The results for five randomly initialized networks are visualized with different colors. TN, TS, and D are the abbreviations for target navigation, target search, and depth, respectively.

Table 1 lists the training time taken for each experiment, with five random initializations conducted for each task. Fig. 2 supplements these results by visually depicting the success rates achieved over training iterations. Each experiment had varying time and iteration, owing to the termination criterion mentioned in the experimental settings, achieving a success rate of 90% or higher for five consecutive iterations.

A review of these baseline results reveals extensive time requirements for training the drone agent via DRL, even within the relatively simple environments depicted in Fig. 1. More specifically, the target search scenario, which realistically omitted the target's position, required training durations of 30.05 h, 19.01 h, and 11.32 h for depth, RGB-D, and RGB sensors, respectively. This indicates that repeated training from scratch, necessitated by task variations, results in excessive time and computational power consumption, thereby challenging the application of DRL in real-world drone-control settings.

Our observations also indicate that the target navigation task, despite the limitations of sensor type (depth only) and neural network architecture, was completed faster than the target search tasks – 3.39, 5.35, and 8.99% of training time for depth, RGB-D, and RGB sensors, respectively. These results suggest that the availability of the target position significantly reduces task difficulty, leading to quicker training termination. Moreover, as noted in the experimental settings, the target object, being a "red" box, is harder to identify with only depth information as opposed to using RGB-D or RGB.

3.4. Experiment II: Evaluation of the proposed method

This subsection describes the results of training the drone agent using the proposed method. As described in Section 2,

Table 1
Training time (s) for baseline.

	Target navigation (with Depth)	Target search (with Depth)	Target search (with RGB-D)	Target search (with RGB)
1st	3,024	155,643	56,363	39,749
2nd	3,439	73,135	64,639	40,655
3rd	3,602	98,812	78,691	37,400
4th	3,842	100,932	76,666	45,597
5th	3,735	112,365	65,883	40,432
Avg	3,528	108,177	68,448	40,766

Table 2
Comparison of training time (s) between the baseline and the proposed method. "Gap" means the ratio between the proposed method and the baseline training time.

	Baseline	Proposed method				Gap
		Phase 1 Data collection	Phase 2 Supervised pre-training	Phase 3 DRL-based fine-tuning	Total	
Target Search with Depth	108,177.17	68.54	1,750.67	20,192.02	22,011.23	79.65% ↓
Target Search with RGB-D	68,448.47	71.37	1,581.59	1,429.09	3,906.08	94.29% ↓
Target Search with RGB	40,766.48	66.75	1,738.97	2,244.08	4,876.13	88.04% ↓

this includes three phases: (1) data collection, (2) supervised pre-training, and (3) DRL-based fine-tuning.

First, we depict the overall time comparison to the baseline in Table 2. We observed 79.65%, 94.29%, and 88.04% reductions in the training time for target searches with depth, RGB-D, and RGB, respectively, meaning that the proposed method is a practical and efficient alternative to training from scratch.

From now on, we will examine the results for each phase of the proposed method.

The data-collection phase involves data collection for the target search task (target task) using a particular stochastic policy (π_ψ) trained for the target navigation task (source task). We collected data for 100 episodes using six rollout workers for each target-search task. Table 3 lists the size and success rate of the collected data and the time required to collect the data for the three target tasks. Compared with the baseline results in Table 1, the time required for data collection listed in Table 2 was marginal. On an average, it took 68.89 s. This tends to be proportional to the number of rollout workers used for data collection. We omitted the time comparison based on the number of rollout workers because of the marginal effect on the total training time of the proposed method and space limitations.

The next phase involves pre-training the target agent using the collected data under supervision. We performed the experiments using five randomly initialized networks. The training epochs, time, policy loss, and value loss averaged over five experiments are listed in Table 4. 1,750.67 s, 1,581.59 s, and 1,738.97 s were taken for pre-training the target task with depth, RGB-D, and RGB, respectively, which were only 1.62%, 2.31%, and 4.27% of the computational time for the baseline. Table 5 presents the evaluation results of five pre-trained target agents for each task over 100 episodes. The

Table 3
Average size and success rate (%) of the collected data, and data collection time (s) in Phase 1.

	Target search (with Depth)	Target search (with RGB-D)	Target search (with RGB)
Size (unit)	21,047	20,421	18,596
Success Rate (%)	90	90	91
Collection Time (s)	68.54	71.37	66.75

average success rates are 82.2%, 76.8%, and 62.6% for the target search with depth, RGB-D, and RGB, respectively. Because the source task used only a depth sensor, we can infer that when pre-training the target task using the data collected by rolling out the source agent, relatively high performance was obtained owing to good alignment with depth, and low performance was obtained owing to poor alignment with RGB.

The final phase involved fine-tuning the target agent in the target task via DRL. Table 6 lists the training times. The success rates along the training iterations are presented in Fig. 3. We observed a significant reduction in the training time and iterations compared with the baseline. This accelerated training is attributed to pre-trained networks containing knowledge transferred from the source agent through phases 1 and 2. The training time for Depth in Phase 3 accounted for approximately 18.7% of the Baseline, whereas RGB and RGB-D required only 12.0% and 0.2%, respectively. This difference is attributed to trial-and-error-based DRL's challenge in locating the 'red' box using a depth image that cannot encode color.

Overall, we can conclude that the proposed method is suitable for real-world drone applications because of its rapid adaptation to task variations by leveraging the knowledge of previously trained agents.

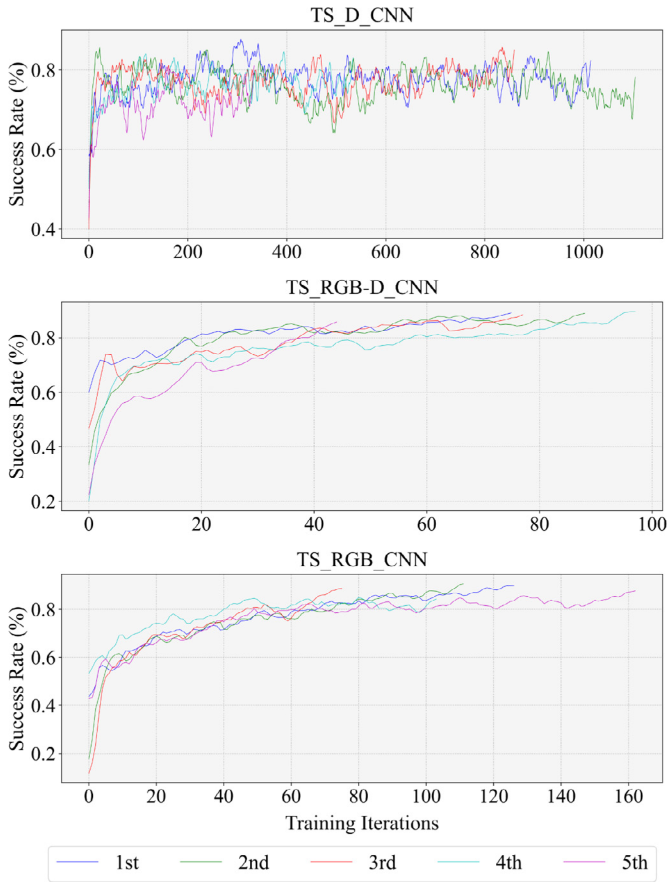


Fig. 3. Success rate (%) over training iteration for Phase 3.

Table 4
Average training epochs, time (s), and losses for policy and value networks for Phase 2.

	Target search (with Depth)	Target search (with RGB-D)	Target search (with RGB)
Epochs (#)	345	315	361
Time (s)	1,750.67	1,581.59	1,738.97
Policy Loss ($\mathcal{L}_p$)	0.013	0.010	0.012
Value Loss ($\mathcal{L}_v$)	0.025	0.013	0.028

Table 5
Success rate (%) of the pre-trained model obtained through Phase 2.

	Target search (with Depth)	Target search (with RGB-D)	Target search (with RGB)
1st	78	79	62
2nd	82	73	64
3rd	82	72	63
4th	84	83	62
5th	85	77	62
Avg	82.2	76.8	62.6

4. Conclusions

In this study, we present a three-phase deep reinforcement learning (DRL) algorithm designed to improve efficiency and flexibility. The proposed approach facilitates knowledge transfer by efficiently collecting training data for an agent on a target task with a trained agent on a source task. This mechanism accelerates the training process under task variation as well as provides flexibility to change the input modality and neural network architecture. Empirical evaluations covering different missions, sensor types, and architectures of drone agents showed a significant reduction in the training time of up to 94.29%. Future research could address the incorporation of real-world drone flight data or even extend the application spectrum to other relevant domains. Considering the frequency of task variations in real-world scenarios, we are optimistic that our approach will advance DRL-driven autonomous drone control.

Table 6
Training time (s) for Phase 3.

	Target search (with Depth)	Target search (with RGB-D)	Target search (with RGB)
1st	26,226.26	2,282.76	3,361.00
2nd	29,934.55	2,540.64	2,905.41
3rd	22,578.86	2,332.48	1,944.75
4th	13,598.53	2,774.80	2,728.98
5th	8,621.92	1,334.94	4,411.92
Avg	20,192.02	2,253.12	3,070.41

CRediT authorship contribution statement

Sooyoung Jang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Hyung-Il Kim:** Conceptualization, Formal analysis, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly supported by Electronics and Telecommunications Research Institute (ETRI), South Korea and Institute for Information & Communications Technology Planning & Evaluation (IITP), South Korea grant funded by the Korea government (MSIT), South Korea (23ZR1140, No. 2022-0-00124).

References

[1] H.X. Pham, H.M. La, D. Feil-Seifer, L.V. Nguyen, Autonomous UAV Navigation using Reinforcement Learning, 2018, arXiv preprint arXiv:1801.05086.
[2] S. Jang, C. Choi, Prioritized Environment Configuration for Drone Control with Deep Reinforcement Learning, Hum.-Cent. Comput. Inf. Sci. 12 (2) (2022) 1–16.

[3] E. Cetin, C. Barrado, G. Munoz, M. Macias, E. Pastor, Drone Navigation and Avoidance of Obstacles through Deep Reinforcement Learning, in: Proceedings of IEEE/AIAA 38th Digital Avionics Systems Conference, DASC, 2019, pp. 1–7.

[4] Y. Song, M. Steinweg, E. Kaufmann, D. Scaramuzza, Autonomous Drone Racing with Deep Reinforcement Learning, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2021, pp. 1205–1212.

[5] W.J. Yun, S. Jung, J. Kim, J.-H. Kim, Distributed deep reinforcement learning for autonomous aerial eVTOL mobility in drone taxi applications, ICT Express 7 (1) (2021) 1–4.

[6] C. Piciarelli, G.L. Foresti, Drone Patrolling with Reinforcement Learning, in: Proceedings of International Conference on Distributed Smart Cameras, 2019, pp. 1–6.

[7] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015, arXiv preprint arXiv:1503.02531.

[8] Z. Gao, K. Xu, B. Ding, H. Wang, KnowRU: Knowledge reuse via knowledge distillation in multi-agent reinforcement learning, Entropy 23 (8) (2021).

[9] W.M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, M. Jaderberg, Distilling policy distillation, in: Proceedings of International Conference on Artificial Intelligence and Statistics, (AISTATS), 2019, pp. 1331–1340.

[10] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, A. Lerchner, DARLA: Improving Zero-shot Transfer in Reinforcement Learning, in: Proceedings of International Conference on Machine Learning, ICML, 2017, pp. 1480–1490.

[11] S. Gamrian, Y. Goldberg, Transfer Learning for Related Reinforcement Learning Tasks via Image-to-Image Translation, in: Proceedings of International Conference on Machine Learning, ICML, 2019, pp. 2063–2072.

[12] X. Chen, Z. Zhou, Z. Wang, C. Wang, Y. Wu, K. Ross, BAIL: Best-action Imitation Learning for Batch Deep Reinforcement Learning, in: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 18353–18363.

[13] K. Arndt, M. Hazara, A. Ghadirzadeh, V. Kyrki, Meta Reinforcement Learning for Sim-to-Real Domain Adaptation, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 2725–2731.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms, 2017, arXiv preprint arXiv:1707.06347.

[15] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous Methods for Deep Reinforcement Learning, in: Proceedings of International Conference on Machine Learning, ICML, 2016, pp. 1928–1937.

[16] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M.I. Jordan, I. Stoica, Ray: A Distributed Framework for Emerging AI Applications, in: 13th USENIX Symposium on Operating Systems Design and Implementation, (OSDI 18), 2018, pp. 561–577.

[17] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, I. Stoica, RLlib: Abstractions for Distributed Reinforcement Learning, in: Proceedings of International Conference on Machine Learning, ICML, 2018, pp. 3053–3062.