Hindawi
Journal of Sensors
Volume 2021, Article ID 6652042, 8 pages
https://doi.org/10.1155/2021/6652042

*Research Article*

# Reinforcement Learning Guided by Double Replay Memory

**Jiseong Han**,[1] **Kichun Jo**,[2] **Wontaek Lim**,[3] **Yonghak Lee**,[1,4] **Kyoungmin Ko**,[1] **Eunseon Sim**,[1] **JunSang Cho**,[5] **and SungHwan Kim**[1]

[1]*Department of Applied Statistics, Konkuk University, Seoul, Republic of Korea*
[2]*Department of Smart Vehicle Engineering, Konkuk University, Seoul, Republic of Korea*
[3]*Department of Automotive Engineering, Hanyang University, Seoul, Republic of Korea*
[4]*AI Analytics Team, Deep Visions, Seoul, Republic of Korea*
[5]*Industry University Cooperation Foundation Konkuk University, Republic of Korea*

Correspondence should be addressed to Wontaek Lim; lwt1849@gmail.com and SungHwan Kim; shkim1213@konkuk.ac.kr

Received 26 November 2020; Revised 7 February 2021; Accepted 24 March 2021; Published 29 April 2021

Academic Editor: Ismail Butun



Experience replay memory in reinforcement learning enables agents to remember and reuse past experiences. Most of the reinforcement models are subject to single experience replay memory to operate agents. In this article, we propose a framework that accommodates doubly used experience replay memory, exploiting both important transitions and new transitions simultaneously. In numerical studies, the deep $Q$-networks (DQN) equipped with double experience replay memory are examined under various scenarios. A self-driving car requires an automated agent to figure out when to adequately change lanes on the real-time basis. To this end, we apply our proposed agent to the simulation of urban mobility (SUMO) experiments. Besides, we also verify its applicability to reinforcement learning whose action space is discrete (e.g., computer game environments). Taken all together, we conclude that the proposed framework outperforms priorly known reinforcement learning models in the virtue of double experience replay memory.

## 1. Introduction

Machine learning is importantly used to address the challenges in an autonomous car of localization, road, and pedestrian detection, for instance, [1] using convolutional neural network with self-supervised learning requiring human road annotations. This research allows to annotate automatically using OpenStreetMap [2]. Kocamaz et al. [3] suggest the vision-based pedestrian and cyclist detection method with the multicue cluster algorithm designed to reduce false alarms. Another important benchmark in the autonomous car is to efficiently control in the real road. Hence, in the autonomous car industry, an effective lane change is one of the most imperative issues to solve. In addition, it is mainly due to the fact that highway traffic accident happens in reality in the middle of lane change. Many cutting-edge techniques are proposed in pursuit of being suited to practical traffic environments [4]. For instance, Yang et al. [5] suggest the adaptive and efficient lane change trajectory planning for autonomous vehicles. In addition, Cesari et al. [6] and Suh et al. [7] focus on the required controller to track the planned trajectory. Of late, much of the contribution on the basis of the collected and examined naturalistic driving data ia aimed at emulating human driving skills in the context of self-driving cars [8–10]. What is more, the latest studies successfully develop the end-to-end learning technique in pursuit of figuring out the relationship between video sensing data and lane change decision [11]. Now that reinforcement learning (RL) has been widely applied to modeling and planning self-driving car, the lane change problems are addressed by use of RL-based agents in various experiments [12–15].

The reinforcement learning is, in theory, designed to maximize numerical rewards by the agent interacting with the environment [16]. It is commonplace that reinforcement learning faces prohibitive computing costs mostly due to high-dimensional data of vision or speech analysis, and thus, a policy in RL hardly adapts to high complexity. In the blessing of recent computing technology, a RL model whose

**Given:** :
An off-policy RL algorithm $\mathbb{A}$, where $\mathbb{A}$: DQN
Sampling strategies ($\mathbb{S}_1$,$\mathbb{S}_2$) from replay
where: $\mathbb{S}_1$ : uniform sampling, $\mathbb{S}_2$ TD-error based sampling
an update probability strategy $S$ for update second replay, where $S$ : $w_t = \delta_t^{\,2}/\sum_n \delta_n^{2.}$
**Initialize** $\mathbb{A}$
**Initialize** replay buffer $H_1$, $H_2$
**observe** S0 and choose a0 using $\mathbb{A}$
**for** episode =1, M **do**
**observe**$(s_t\,; r_t\,; s_{t+1}\,; p_t\,; H_1)$
store transition $(s_t, a_t, r_t, s_{t+1}\,; p_t\,; H_1)$ in $H_1$ to follow $S_1$
**for** t=1; T **do**
**if** N2>k **then**
With $\mathbb{S}_1$,$\mathbb{S}_2$, sampling ratio $\lambda$
sample transition from $H_1$ and $H_2$
**else**
With $\mathbb{S}_1$, sample transition from H1
update weight according to $\mathbb{A}$
put used transition into $H_2$ with probability $p_t\,; H_2$
**if** transitions from $H_2$**then**
update $p_t\,; H_2$ according to $S$
**until** converge

Algorithm 1: Double experience replay (DER).

Table 1: The scores from CartPole simulations.

| | Max score | Average score |
|---|---|---|
| DQN | 373.80 | 229.30 |
| $\lambda = 0.1$ | 478.65 | 210.58 |
| $\lambda = 0.5$ | 500 | 259.23 |
| $\lambda = 0.9$ | *500* | *285.49* |
| PER | 500 | 237.63 |

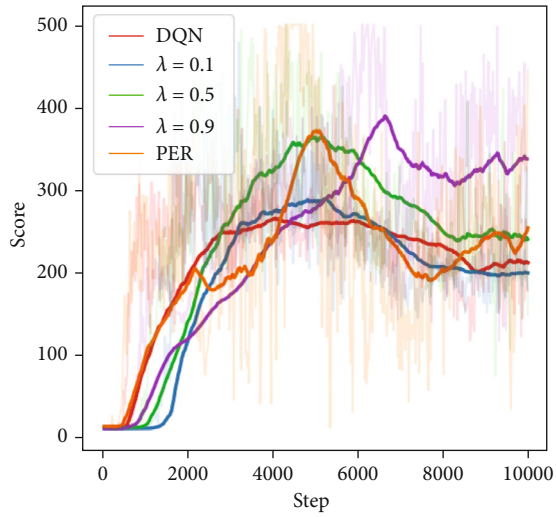
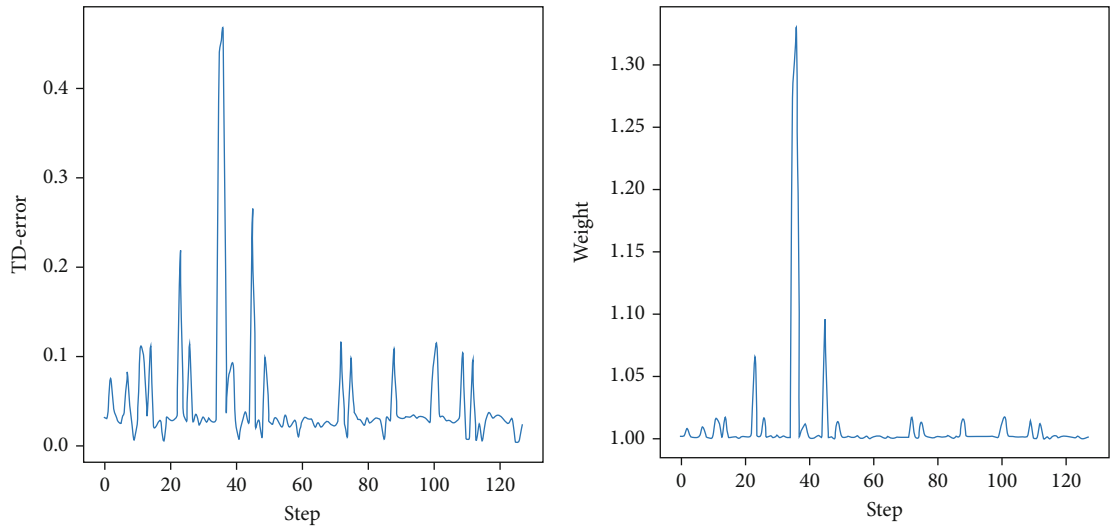



Figure 1: Simulation results via CartPole to compare to DQN and PER.

policy learns on deep learning is known to be efficient to approximate policy and thereby to dramatically improve applicability to diverse environments (e.g., deep $Q$-learning (DQN) [17]). Experience replay method plays a significant role in deep $Q$-learning, allowing an agent to remember and reuse past experiences. This method functions to enhance the use of data and attenuates strong correlation between samples. The current off-policy algorithm based on experience replay adopts only uniform sampling such that transitions are sampled at an equal chance. In this regard, these approaches only focus on the strong correlation between samples. Contrary to this, the rule-based replay sampling has been introduced. For instance, prioritized experience replay (PER) builds on temporal-difference (TD) error and is well known to improve the deep $Q$-network under the Atari environment [18]. Besides, the recent PER-type method [19] adopts different sizes of the replay memory, remembers, and forgets experience memory in part in order to improve performance to a large extent. And yet, these methods are limited in a scope to single experience replay memory. Lately, both optimizing hyperparameter and the safe learning without any assumptions about model dynamics have been actively studied in reinforcement learning fields. This is because finding the optimal hyperparameter needs repetitive experiments and it typically costs expensive reinforcement learning tasks. Dong et al. [20] use the Siamese-based correlation filter-based method to optimize hyperparameters. Liu et al. [21] suggest methods that the robust reinforcement model exploits the ensemble method to accommodate model dynamics and robust crossentropy methods to optimize the control sequence with constraint functions.

In this paper, we proposed a novel method called double experience replay memory (DER), which facilitates to sample transitions efficiently and to exploit replay memory simultaneously. More precisely, we adapt our method with uniform




(a) TD-error (b) Sample weight

Figure 2: Absolute value of TD-error (a) and sample weight (b) in CartPole.

sampling and TD-error-based sampling methods using the hyperparameter. To verify its practical utility, this proposed method is assessed under a range of experiment scenarios.

## 2. Related Works

Simply put, the deep $Q$-networks (DQN) [17] train an agent for lane changes of discrete action space. The prioritized experience replay (PER) [18] improves the DQN with prioritization. Here, we briefly go over DQN and PER in the algorithmic standpoint before we give accounts for the proposed algorithm.

*2.1. Deep Q-Networks (DQN).* The goal of reinforcement learning (RL) is aimed at finding the policy that maximizes rewards [16]. Typically, RL iteratively updates $Q$-function based on $Q$-learning but RL suffers from several challenges. First, in order to simulate real world, a countless number of states are inevitably required. Secondly, correlation between samples is intensively high in common. To tackle with this, the deep $Q$-networks (DQN) [17] pioneered to adopt deep learning reinforcement learning algorithm, where the DQN replaces $Q$-table with neural network. The $Q$-network predicts the reward in numerous real-world states and stores and samples data in the experience replay (Lin, 1992) so that it reduces sample correlation. Over the years, many have proposed variants of DQN to improve. For instance, the NoisyNet-DQN (Fortunato et al. 2017) added parametric noises to weights based on the DQN structure and gained higher scores in Atari games. The ensemble-DQN (Chen et al. 2018) developed the ensemble network for deep reinforcement learning. Furthermore, the Random Ensemble Mixture (REM) (Agarwal et al. 2019) proposed the offline $Q$ -learning algorithm and proved that the algorithm can lead to high-quality polices using DQN-based experiments. Lastly, the NROWAN-DQN (Han et al. 2020) suggested a noise reduction method for NoisyNet-DQN and designed weight adjustment strategy. In this regard, it is confirmed that the DQN contributes to significantly advancing the RL domain.

Table 2: The scores from Atari simulations.

| | DQN | $\lambda = 0.1$ | $\lambda = 0.5$ | $\lambda = 0.9$ | PER |
|---|---|---|---|---|---|
| River Raid | 1437.80 | 1404.10 | 1294.80 | 920.30 | 962.20 |
| Space Invaders | 292.00 | *671.15* | 420.35 | 99.40 | 445.50 |
| Boxing | 25.03 | *40.42* | -18.19 | 0.49 | 7.29 |
| Breakout | 52.86 | *58.28* | 44.31 | 4.96 | 31.14 |

The deep $Q$-network (DQN) [17] is known as a model-free reinforcement learning (RL) algorithm for discrete action space. The DQN updates parameters of the $Q$-network in order to derive an approximated $Q$-value, where $Q$ is defined as $\pi_Q(s) = \text{argmax}_{a \in A} Q(s, a)$, where $Q(s_t, a_t) = \mathbb{E}[\sum_{k=0}^{\inf} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a]$. A greedy policy facilitates to search an optimal $Q$-value. We can invite in RL models an $\varepsilon$-greedy policy related to $Q$, a policy controlled by a probability $\varepsilon$ that determines a random action (uniformly sampled out of action space) and takes the action $\pi_Q(s)$ with probability $1 - \varepsilon$. In the process of training, an agent explores episodes subject to $\varepsilon$-greedy policy on the basis of the current approximation of the action-value function $Q$ [16]. The transition tuples $(s_t, a_t, r_t, s_{t+1})$ as a by-product are generated and stored in memory (a.k.a. replay buffer), where $s_t, a_t, r_t$ are a state, action, and reward at time $t$, respectively. The $Q$-network learns on the Bellman equation:

$$L = \mathbb{E}(Q(s_t, a_t) - y_t)^2, \text{ where } y_t = r_t + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}). \tag{1}$$

Transitions are stored as $(s_t, a_t, r_t, s_{t+1})$ tuple into replay buffer, and this tuple was sampled from derived replay buffer uniformly. This replay memory attenuates correlation across consecutive states on account of vast samples.

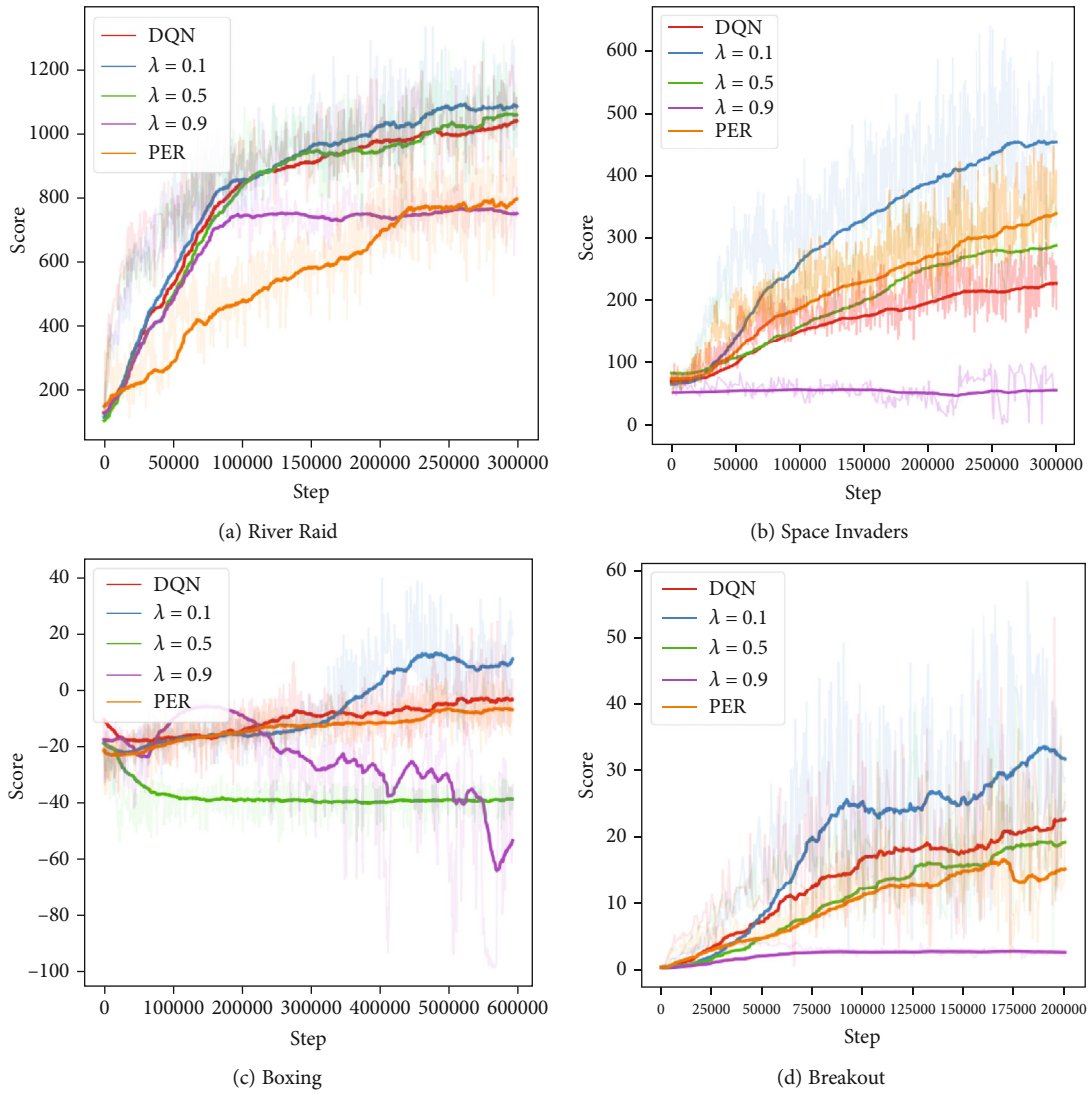



Figure 3: Simulation results via Atari games to compare to DQN and PER.

*2.2. Prioritized Experience Replay (PER).* Former reinforcement learning models are designed to uniformly sample from experience replay with no consideration of the degree of transition importance. The idea behind the prioritized experience replay (PER) [18] is to differently sample from distribution subject to artificial environments (e.g., excessively good or poor performance). To update an action-value $Q(s, a)$, we adopt the TD-error as loss to update approximated action-value function in place of $Q(s, a)$ as follows:

$$\delta = r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t). \quad (2)$$

The value of the TD-error, in theory, measures a rate that agent learns from the experience. Precisely, the high absolute TD-error means that correction for the expected action value function becomes large. Experiences with high TD-error relate to good performance in episodes. On the contrary, experiences with large negative TD-error are associated with poor performance in episodes. It is shown that this artificially designed sampling scheme is superior to improve agents on the whole. Interestingly, it is noteworthy that the Prioritized Sequence Experience Replay (PSER) (Brittain et al. 2019) proposed a framework for prioritizing sequences of experience to learn efficiently. PSER not only assigned high priorities to important experiences like PER but also propagated the priorities to previous experiences that lead to important experiences. It considers the sequence importance by increasing the earlier experience priority. Importantly, this method is featured with selective experiences to improve accuracy.

## 3. Proposed Algorithm

Here we propose the reinforcement learning model, called as the double experience replay memory (DER), that builds on an combination of multiple sampling transitions. In what follows, the Algorithm 1 is described: First, we separate two replay memories $H_1$ and $H_2$, each consisting of state, action, reward, and sampling probability denoted by $s_t, a_t, r_t$, and $p_t$, respectively. Second, an agent experiences repeated episodes and stores transition $(s_t, a_t, r_t, s_{t+1}, p_{t,H_1})$ in $H_1$, where $p_{t,H_1}$ is a

(a) Ring network (b) Yeongdong Bridge (c) Yeongdong Bridge in SUMO

Figure 4: The illustration of SUMO environments to simulate the (a) ring network and (c) urban traffic network and (b) its satellite image.

weight elements at $t$ time step where $t = 1, 2, 3, \cdots$ and is assumed to follow an arbitrary distribution. With a little bit of episodes, an agent learns based on transitions sampled from $H_1$. Subsequent to this, transitions move to $H_2$ with another weight $p_{t,H_2}$ which follows a predefined distribution so that we reuses these transitions sampled from $H_1$. In other words, $H_2$ is equivalent to transitions sampled from $H_1$ used to construct models. When $H_2$ complies an adequate amount of transitions to train, we sample from both $H_1$ and $H_2$ alternately with the parameter $\lambda$, where $\lambda \in [0, 1]$ is a constant of adjusting the selecting batch data ratio between $H_1$ and $H_2$. For example, if $\lambda = 0.1$, it means taking 90% of train data from $H_1$ and only 10% of data from $H_2$. Both sampling probabilities within replay memory (i.e., $p_{t,H_1}$ and $p_{t,H_2}$) are updated via the predefined rule. Importantly, sampling probability (i.e., $p_{t,H_1}$ and $p_{t,H_2}$) determines what transitions are chosen, while selection frequency (i.e., $\lambda$) determines the ratio of replay memory between $H_1$ and $H_2$.

*3.1. Uniform and TD-Error-Based Weight.* In this section, we describe an example of double experience replay memory. In $H_1$, we use uniform sampling strategy, and in $H_2$, we use the TD-error- (i.e., $\delta_t$) based sampling strategy inspired by prioritized experience replay (PER). We uniformly sample transitions in $H_1$ such that $p_{t,H_1} = 1/(\text{buffer size})$. In contrast, the TD-error-based sampling strategy applies to $H_2$ as follows: $p_{t,H_2} = e^{|\delta_t|}$. In principle, in order to make weight to 1 for frequently sampled transition, we set the initial value to exponential $e$ and update weights as follows:

$$p_{t,H_2} \leftarrow \left(p_{t,H_2}\right)^{w_t}, \text{ where } w_t = {\delta_t}^2 / \sum_n \delta_n^2. \tag{3}$$

Mathematically, the transitions with the large TD-error are sampled with high chance and the frequently sampled transitions converge to 1 which is the baseline and the lowest value for the sampling portion. $|\delta_t|$ never goes to negative value and $w_t \leq 1$, and so the repetition of exponential $(p_{t,H_2})^{w_t}$ converges to 1. It is important to note that $w_t$ adjusts the sampled transitions in a balanced way because $w_t$ reduces the chance of transitions if they are chosen at preceding steps.

Table 3: The scores from SUMO simulations.

| | Ring network | Yeongdong Bridge |
|---|---|---|
| DQN | 135.74 | 60.64 |
| $\lambda = 0.1$ | 117.46 | 75.84 |
| $\lambda = 0.5$ | 122.78 | 70.25 |
| $\lambda = 0.9$ | *216.91* | *81.77* |
| PER | 135.71 | 79.50 |

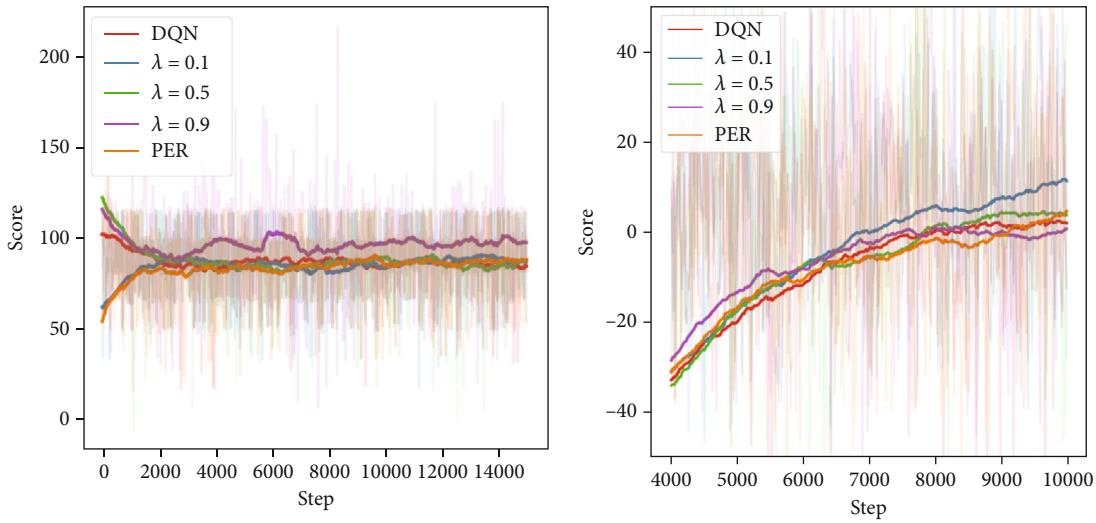
Related to a network architecture, we use the deep-$Q$ network (DQN) as a baseline algorithm. The proposed method utilizes both transition sampling strategies of uniformly sampling in the DQN and TD-error-based sampling strategy. In this regard, this predefined rule can be viewed as an intermediate-type method between the two reinforcement models. In principle, this, if $\lambda$ equals to 0, means uniformly sampling from DQN, and if $\lambda$ equals to 1, this is the same as sampling only from the TD-error-based strategy. Putting together all strategies in a single view, Figures S2 and S3 describe the algorithm pipeline.

## 4. Numerical Experiments

Without a loss of generality, we first evaluate if the proposed methods are flexibly applicable to diverse computer game environments, and these are followed by autonomous car experiments.

*4.1. CartPole-v1.* Below, we conduct experiments based on the CartPole environment provided by the Open AI gym [22]. We use the multilinear perceptron model of 256 cells. We apply the Adam algorithm [23] and set parameters (e.g., learning rates = 0.0005, $\gamma = 0.98$, batch size = 128, $N_1$ = 50,000, $N_2$ = batch size × 100, where $N_1$ and $N_2$ denote the buffer size of $H_1$ and $H_2$, respectively). We train the model with 10,000 steps.

To compare result between ratios of memory, we use $\lambda$ as 0.1, 0.5, and 0.9. Within the DQN as a baseline algorithm, we only change experience replay memory. We compute the max score of average of returns for 20 episodes. To compare,




(a) Ring network

(b) Yeongdong Bridge

Figure 5: Simulation results via SUMO to compare to DQN and PER.

average scores across all episodes are presented in the sense that CartPole rapidly reaches the max score. In Table 1 and Figure 1, we observe that the proposed model performs better when $\lambda$ increases in size. As a result, this clearly shows that the proposed method performs better than uniformly sampled experience memory (DQN) and PER when $\lambda = 0.9$. Interesting, Figure 2 shows the optimally derived TD-error and weight, implicating that TD-error and weight are stabilized as iterated.

## 5. Atari

Atari is the video game environment in which the reinforcement learning can be applied and uses vision data as input [17]. In what follows, we specify experiment configurations. The DQN builds on convolution neural network, whose input is composed of $84 \times 84 \times 4$ with 4 stacked frames. We resize image, convert into grey scale, and normalize input data. The first layer has 32 filters of $8 \times 8$ with strides 4 and applies a rectifier nonlinearity unit (ReLU) function. The third layer has 64 filters of $4 \times 4$ with strides 2 and also applies the ReLU function. The final layer has 64 of $3 \times 3$ with stride 1 followed by the ReLU function. The last layer is fully connected that consists of 512 ReLU. The output layer is fully connected with all available actions (see Figure S1). Regarding the set of parameters, we use as default the Adam algorithm with learning rate $= 0.0005$, $\gamma = 0.99$, $N_1 = 50{,}000$, batch size $= 128$, and $N_2 =$ batch size $\times 100$, where $N_1$ and $N_2$ denote the buffer size of $H_1$ and $H_2$, respectively. To compare performance, we compute average values by 100 episode returns and max values, respectively. We proceed for adequate learning with more than 200,000 steps. In Table 2 and Figure 3, we found that the proposed method with $\lambda = 0.1$ obtains the best score for Space Invaders, Boxing, and Breakout. Taken together, in many Atari environments, we found the DER performs better than the DQN with uniform sampling and PER.

*5.1. Urban Mobility.* The simulation of urban mobility (SUMO) is an open-source simulation package designed to simulate urban traffic network [24]. The SUMO provides simple networks, creates user-defined networks, and allows real-urban simulations using OpenStreetMap (OSM; OpenStreetMap contributors [2]). The SUMO facilitates to assess traffic-related problems such as a traffic light control, route choice, and self-driving car simulation. In addition, the SUMO supports the Python API with TraCI [25], making it possible to evaluate by each time unit. In this paper, we create the ring network environment and hypothesize whether a self-driving car effectively changes lane or not. The following is the proposed simulation schemes. To begin, we consider two rings on which each vehicle moves around. At the outset, the agent vehicle (i.e., maneuvered from RL rules) is placed in the outside ring and keeps moving around. The agent determines the moment to change a lane, thereby pushing towards the inner circle without collision as in Figure 4(a). In light of reward, we impose as baseline the logarithm of average speed across all working vehicles, aiming at no traffic jam given a ring network. Precisely, if an agent changes a lane successfully, we add 100 for reward, whereas we bring back 100 if the agent causes collision with another vehicle. Under this simulation environment, we take only into account the lane change for simplicity. The essentials in driving such as acceleration, brake, and steering are automatically maneuvered by SUMO priorly optimized system. For each state, an agent is allowed to determine the agent's vehicle speed, while other neighboring vehicle's speed is with regulation that distance is limited within 30 meters out of the agent vehicle. To verify the advantages of the proposed model, we compare DQN, PER, and our method with $\lambda$ ranging from 0.1 to 0.9 and iteration for train is made 15,000 steps. In addition to this, we create a network via OSM, which emulates the real district nearby Yeongdong Bridge located in Seoul, South Korea. Figure 4(b) describes the configuration of maps. In this simulation, we focus on the lane change performance, whose environment factors are identical to the ring network

scenario and follow rule-based acceleration and brake provided in the SUMO as default other than options for lane change decision.

Table 3 includes the total reward that each model produces. Importantly, the proposed method (DER) is superior to PER in reward (e.g., ring network: 216.91 (DER) and 135.71 (PER) for $\lambda = 0.9$, Yeongdong Bridge: 81.77 (DER) and 79.50 (PER) for $\lambda = 0.9$). It is clear that our method dominantly outperforms DQN and PER. More importantly, we observe that the large $\lambda$ increases reward scores (see Table 3 and Figure 5).

## 6. Discussion

This paper proposes the double experience replay (DER) that accommodates two different replay memories in order to train an agent with important transitions and newly explored transitions simultaneously. Here, we predefine the diminishing weight rules to decrease bias in place of important sampling methods like the PER. In simulations, we compare this method with uniform distribution and prioritized replay memory (PER) using temporal-difference (TD) error and find out that the DER performs better in various environments implemented by OpenAI gym. Besides, an agent vehicle in the SUMO environment is also found to effectively change the lanes. Interestingly, the simulations of SUMO and CartPole show that transition with the high absolute TD-error is suited to short and repeated episodes. It is also worth to develop a benchmark to determine the size of each buffer for occupying adequate memory size and improve computation time in an algorithmic context to advance its applicability. Recent papers suggest various methods motivated from the replay memory. The selective experience replay for lifelong learning [26] determines what experiences to store. They complement FIFO buffer on the basis of reward-based and global distribution matching strategies. On the other hand, experience replay optimization (ERO, Zha et al. [27]) proposed two polices that one updates the agent policy and the other updates the replay policy. The former updates to maximize the cumulative reward, and the latter updates to provide useful experiences to the agent (see Figure S4). The competitive experience replay exploits the relabeling technique to fit an agent in a sparse reward environment. The relabeling technique is known to accelerate performance. In future research, we can apply this method with the DER simultaneously in sparse reward environments.

## Data Availability

We uploaded download link in publication part at our website (http://www.hifiai.pe.kr).

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Authors' Contributions

Jiseong Han and Kichun Jo are co-first authors.

## Acknowledgments

This research was supported by Konkuk University Researcher Fund in 2019, Konkuk University Researcher Fund in 2020, and the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2020R1C1C1A01005229, 2020R1C1C1007739, and 2019R1I1A1A01061824).

## Supplementary Materials

Figure S1: the convolution neuron network architecture for reinforcement learning. Figure S2: the flow chart of DER. Figure S3: process of sampling transitions with ratio $r$ and update weight by predefined rule. Figure S4: simulation results via CartPole to compare to the DQN and ERO *(Supplementary Materials)*

## References

[1] A. Laddha, M. K. Kocamaz, L. E. Navarro-Serment, and M. Hebert, "Map supervised road detection," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 118–123, Gothenburg, Sweden, 2016.

[2] OpenStreetMap contributors, *Planet Dump*, 2017, https://planet.osm.org.https://www.openstreetmap.org.

[3] M. K. Kocamaz, J. Gong, and B. R. Pires, "Vision-based counting of pedestrians and cyclists," in *2016 IEEE winter conference on applications of computer vision (WACV)*, pp. 1–8, Lake Placid, NY, USA, 2016.

[4] J. Nilsson, M. Brannstrom, E. Coelingh, and J. Fredriksson, "Lane change maneuvers for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1087–1096, 2016.

[5] D. Yang, S. Zheng, W. Cheng, P. J. Jin, and B. Ran, "A dynamic lane-changing trajectory planning model for automated vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 228–247, 2018.

[6] G. Cesari, G. Schildbach, A. Carvalho, and F. Borrelli, "Scenario model predictive control for lane change assistance and autonomous driving on highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 23–35, 2017.

[7] J. Suh, H. Chae, and K. Yi, "Stochastic model-predictive control for lane change decision of automated driving vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 4771–4782, 2018.

[8] L. Li, C. Lv, D. Cao, and J. Zhang, "Retrieving common discretionary lane changing characteristics from trajectories," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2014–2024, 2017.

[9] Q. Wang, Z. Li, and L. Li, "Investigation of discretionary lane-change characteristics using next-generation simulation data sets," *Journal of Intelligent Transportation Systems*, vol. 18, no. 3, pp. 246–253, 2014.

[10] D. Xu, Z. Ding, H. Zhao, M. Moze, F. Aioun, and F. Guillemard, "Naturalistic lane change analysis for human-

like trajectory generation," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1393–1399, Changshu, China, 2018.

[11] S.-G. Jeong, J. Kim, S. Kim, and J. Min, "End-to-end learning of image based lane-change decision," in *2017 IEEE intelligent vehicles symposium (IV)*, pp. 1602–1607, Los Angeles, CA, USA, 2017.

[12] L. Fridman, J. Terwilliger, and B. Jenik, "Deeptraffic: crowd-sourced hyperparameter tuning of deep reinforcement learning systems for multi-agent dense traffic navigation," 2018, https://arxiv.org/abs/1801.02805.

[13] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1379–1384, Changshu, China, 2018.

[14] P. Wolf, K. Kurzer, T. Wingert, F. Kuhnt, and J. M. Zollner, "Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 993–1000, Changshu, China, 2018.

[15] C. You, J. Lu, D. Filev, and P. Tsiotras, "Highway traffic modeling and decision making for autonomous vehicle using reinforcement learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1227–1232, Changshu, China, 2018.

[16] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135, MIT press, Cambridge, 1998.

[17] V. Mnih, K. Kavukcuoglu, and D. Silver, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, https://arxiv.org/abs/1511.05952.

[19] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 316–321, Banff, AB, Canada, 2017.

[20] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, and F. Porikli, "Dynamical hyperparameter optimization via deep reinforcement learning in tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, p. 1, 2019.

[21] Z. Liu, H. Zhou, B. Chen, S. Zhong, M. Hebert, and D. Zhao, "Safe model-based reinforcement learning with robust cross-entropy method," 2020, https://arxiv.org/abs/2010.07968.

[22] G. Brockman, V. Cheung, L. Pettersson et al., "Openai gym," 2016, https://arxiv.org/abs/1606.01540.

[23] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[24] D. Krajzewicz, G. Hertkorn, C. Rossel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th Middle East Symposium on Simulation and Modelling*, pp. 183–187, 2002.

[25] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J. P. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th Communications and Networking Simulation Symposium (CNS'08)*, pp. 155–163, Ottawa, Canada, April 2008.

[26] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[27] D. Zha, K.-H. Lai, K. Zhou, and X. Hu, "Experience replay optimization," 2019, https://arxiv.org/abs/1906.08387.