Optimal Policy Replay:A Simple Method To Reduce Catastrophic Forgetting In Target Incremental Visual Navigation

1st Xinting Li
dept. Northwestern Polytechnical University
Xi'an, China
2021262853@mail.nwpu.edu.cn

2nd Shizhou Zhang
dept. Northwestern Polytechnical University
Xi'an, China
szzhang@nwpu.edu.cn

3rd Yue LU
dept. Northwestern Polytechnical University
Xi'an, China
luyue163@126.com

4th Kerry Dang
dept. Northwestern Polytechnical University
Xi'an, China
kerry357@mail.nwpu.edu.cn

5th Lingyan Ran
dept. Northwestern Polytechnical University
Xi'an, China
lran@nwpu.edu.cn

6th Peng Wang
dept. Northwestern Polytechnical University
Xi'an, China
peng.wang@nwpu.edu.cn

6th Yanning Zhang
dept. Northwestern Polytechnical University
Xi'an, China
ynzhang@nwpu.edu.cn

***Abstract*—Visual navigation is a critical task in robotics and artificial intelligence. In recent years, reinforcement learning-based approaches have gained popularity for visual navigation. However, existing methods lack flexibility in learning multiple navigation targets and suffer from catastrophic forgetting. To address these challenges, we propose a novel paradigm called "target incremental visual navigation" and introduce a method called Optimal Policy Replay (OPR). Target incremental visual navigation aims to study the performance of visual navigation in continuous learning of navigation targets. OPR enables continuous learning of navigation targets without the need for relearning all targets. Our method divides the learning process into on-policy and off-policy stages and stores only the optimal experiences in memory. Experimental results show that OPR effectively alleviates catastrophic forgetting and achieves good performance with a small memory size.**

***Index Terms*—visual navigation, catastrophic forgetting, reinforcement learning, continual learning**

I. Introduction

Visual navigation is a fundamental problem in robotics and artificial intelligence. The target-driven visual navigation task involves commanding an agent to search for a given object in a 3D scene, using its egocentric camera to navigate around obstacles and determine the next step. In recent years, visual navigation has attracted increasing research interest in the fields of artificial intelligence and computer vision, with numerous potential applications such as automated home services, warehouse management, and the hotel industry.

Traditional approaches rely on map-based visual navigation methods. These methods explicitly decompose the navigation task into a set of sub-tasks, including mapping, localization, planning, and motion control [1]–[4], [21]. Due to the recent success of reinforcement-learning-based methods in robotic tasks [22]–[25], many mapless visual navigation works based on reinforcement learning have been proposed [7]–[12]. These methods typically take visual information and the navigation target as inputs and output the optimal actions, and the agent should take at each time step to achieve the specified target. Unlike traditional methods, reinforcement-learning-based approaches directly infer solutions from the current input, which is an end-to-end approach. As such, they require minimal manual engineering and serve as the foundation for new AI-driven visual navigation tasks.

However, current reinforcement-learning-based visual navigation methods typically employ training approaches that randomly select a navigation target at the beginning of each training task when learning multiple navigation targets. In this approach, the network model is capable of learning multiple navigation targets simultaneously. But when the agent is to expand the scope of navigation targets, existing methods must mix new and old goals and relearn. This process not only leads to significant resource wastage but also restricts the applicability of learning-based navigation algorithms. Hence, the model requires the capability of continuous learning of navigation targets. To investigate this matter, we propose a novel visual navigation paradigm known as 'target incremental visual navigation.' In this paradigm, we enable the agent to learn navigation targets continuously in a predefined order, rather than randomly acquiring them.

To address the problem, we introduce a novel method called OPR (Optimal Policy Replay). OPR enables continuous learning of navigation targets without the need for relearning all targets, thereby mitigating resource wastage and expanding the potential applications of reinforcement-learning-based navigation algorithms.This method uses on-policy to learn the current navigation target, uses off-policy to learn only the learned navigation target, and stores only the optimal episode

2023 China Automation Congress (CAC) | 979-8-3503-0375-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/CAC59555.2023.10450433

in memory to ensure the efficiency of off-policy. We found that the OPR method can effectively alleviate the catastrophic forgetting of visual navigation.

In summary, our main contributions are as follows:

1.We introduce a novel paradigm called "target incremental visual navigation" to address the challenge of visual navigation.

2. We introduce a novel framework OPR for target incremental visual navigation, which divides the algorithm into on-policy learning and off-policy learning. Additionally, the memory stores only the optimal policy episode. OPR effectively alleviates catastrophic forgetting in visual navigation.

3.OPR also has well performance with a small memory, indicating its broad applicability in various scenarios.

II. Related Worked

A. Visual Navigation

Visual navigation is one of a fundamental problems for mobile robots. Traditional navigation methods typically use environmental maps for navigation and divide navigation tasks into three steps: mapping, localization, and path planning [1]–[6]. With the development of reinforcement learning, reinforcement learning has been applied to robot tasks, and navigation methods based on reinforcement learning are popular because they solve complex tasks through end-to-end methods. Since Zhu et al [7]. propose an end-to-end navigation model based on deep learning, which implicitly integrates localization, mapping, exploration, and semantic recognition, target-driven visual navigation has developed rapidly, and many efficient models have been proposed. Wortsman et al. propose a meta-learning based method [8] to dynamically adjust the navigation policy according to changes in the environment, and the agent learns self-supervised interaction losses to perform effective navigation. Lee et al. propose object relation graphs to learn spatial relationships between the classes that appear in navigation to better guide agents in navigation [9]. They further propose a novel Visual Transformer network (VTNet) to extract information feature representations in navigation [10]. This information feature representation not only encodes the relationship between objects, but also establishes a strong correlation with the navigation signal, which can better guide the agent's next action.

In the task setting of multiple navigation goals, the above tasks randomly select navigation tasks before the start of each task, which can avoid catastrophic forgetting, but this method greatly limits the flexibility of the model. In actual situations, the model needs to have the ability of target incremental visual navigation.

B. Continual Reinforcement Learning

The issue of catastrophic forgetting in neural networks has gained significant recognition, where after training on the current task and directly training on the next task, the model exhibits high recognition accuracy on the new task but significant degradation in recognition accuracy on the learned tasks. In recent years, people have renewed interest in overcoming catastrophic forgetting in RL. Kirkpatrick et al. proposed the Elastic Weight Consolidation (EWC) method [13], which restricts important weights from past tasks to change more slowly when learning new tasks. Rolnick et al. proposed the Continual Learning with Experience and Replay (CLEAR) method [14], which uses v-trace importance sampling to prevent catastrophic forgetting. Atkinsonf et al. proposed the Reinforcement Pseudo-Rehearsal (RePR) method [15], which generates pseudo-samples from a generative model to maintain knowledge about past tasks in the model. Kessler et al. proposed the UNcertainty guided Continual Learning (UNCLEAR) method [16], which preserves past knowledge by retaining the parameters of the output linear layer. Fernando et al. proposed the PathNet method [17], which uses genetic algorithms to find a path from the input to the output for each task in the neural network, and separates the network parts used at the parameter level from the new task training.

Most of the existing methods based on experience replay use FIFO or reservoir sampling to store experience. These methods cannot cope with the task of target incremental visual navigation very well, because of the limited memory size, they will forget past experience, or affect the learning of new target due to a large number of suboptimal experiences. Therefore, we only store the optimal experience in memory to ensure the efficiency of the model in off-policy.

III. Method

A. Problem Formulation

A navigation task consists of a scene S, an initial point p, and a target object o. The agent's objective is to find the target object o in the 3D environment from the initial position within a given number of steps. The agent's action space is limited to six actions: MoveAhead, RotateLeft/RotateRight, LookDown/LookUp, Done. At each step, the agent receives an egocentric RGB image s from the scene and a target object o and the agent select an action from action space. A collection of all the steps from the beginning to the end of a task is called an episode.The agent successfully completes the navigation task if it performs the "done" action when within 1 meter of an instance of the target object class and within the agent's field of view.

Follw the setting of the target incremental visual navigation, we sequentially provide the target object to be learned. Once a target object has been learned, it will not be re-learned when learning subsequent targets.

B. Method Overview

The OPR utilizing new experiences to learn the policy for the current target and replayed experiences to learn the policy for the learned target. Unlike typical algorithms based experience replay, we only store the optimal experiences in the memory to ensure that the agent can review the optimal policy for learned navigation targets when learning new ones. We train a state feature extraction network and a value-policy network [9]. The input of the value-policy network is the output of the state feature extraction network, which is the

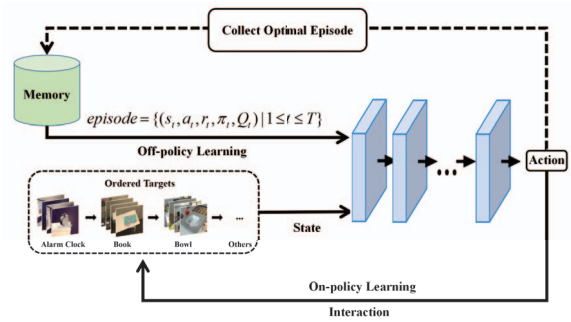

Fig. 1. Algorithm flow overview: The OPR algorithm consists of two stages, namely the on-policy learning stage and the off-policy stage. During the on-policy stage, the agent interacts directly with the environment to learn the policy for the current navigation target. In the off-policy stage, the agent learns the optimal policy for the learned target by utilizing the episode stored in memory. The solid lines in the figure depict the steps executed during each training iteration, while the dashed lines represent the steps executed after a navigation target has been learned.

high-level feature map of the current image. The outputs of the whole network is two functions: policy function $\pi(a|s)$ and Q-function $Q(s|a)$. The entire training process is divided into two stages: the on-policy stages and the off-policy stages. In the on-policy stages, the agent learns the policy for the new target through interaction with the environment, while in the off-policy stages, the agent reviews the learned policy using the optimal experiences of the learned navigation targets stored in the memory to prevent catastrophic forgetting.

C. On-policy learning

In this stage, our objective is to train the agent to learn the optimal policy for the current target.When learning a new navigation target policy, the agent directly interacts with the environment, Training proceeds as in [19] by the A3C algorithm and the policy gradient is given by:

$$G_{on-policy} = \sum_{t=1}^{T_n} \left(Q_\theta^\pi(s_t, a_t) - V_\theta^\pi(s_t)\right) \nabla log \pi_\theta(a_t|s_t) \quad (1)$$

where θ is the parameters of the neural network,$\gamma \in [0, 1)$ is discount factor, $\pi_\theta(a_t|s_t)$ is (current) policy, $Q_\theta^\pi(a_t|s_t)$ is an estimate of action-value function, $V_{\theta(s_t)}^\pi = \sum_{i=0}^{k} \pi_\theta(a_t|s_t) Q_\theta^\pi(a_t|s_t)$is an estimate of the value function, k is the number of action.The pseudocode of on-policy flow is shown in algorithm 1. The agent chooses an initial state and a navigation target at the beginning of each time task, subsequently interacts with the environment during task execution until agent selects the action Done or reaching the maximum step, and updating the network based on the episode once the task concludes.

D. Off-policy learning

In the off-policy stage, unlike the common off-policy algorithm, we do not collect episode during on-policy learning. Instead, we collect episode after the training when a navigation target reaches its maximum. In this way, the collected

Algorithm 1 OPR:on-policy

Reset gradients $d\theta \leftarrow 0$
Initialize parameters $\theta^{'} \leftarrow \theta$
get initial state x_0 and navigation target o.
while not select done or not exceed the maxinum step **do**
  get action $a_t = \pi(s_t|\theta^{'})$ and reward $r_t = R(\cdot|s_t, a_t)$
  get Q function $Q(s_t, a_t)$ and next state $s_{i+1} = f(\cdot|a_t)$
end while
update network parameter $\theta \leftarrow \theta + \alpha G_{on-policy}$

experience is the optimal experience for the current navigation target (assuming that the success rate of the task increases with the number of training times), which can minimize forgetting. Episode stored in memory, $episode = \{(s_t, a_t, r_t, \pi_t, Q_t)|1 \leq t \leq T\}$, $target = (target_1, target_2, ..., target_n)$, T is total step of episode, m is the memory size, The expression of memory is as follows:

$$Memory = \sum_{i=1}^{n} \frac{m}{n} \sum_{j=1}^{T} (s_t, a_t, r_t, \pi_t, Q_t) \quad (2)$$

From a policy perspective, what is stored in memory is:

$$Memory = (\pi_1, \pi_2, \pi_3, \ldots, \pi_n) \quad (3)$$

The off-policy stage occurs several times after the on-policy stage has ended. Training proceeds as in [20] by the ACER off-policy learning algorithm, which use Retrace $Q^{ret}(a_t|s_t)$ estimate $Q^\pi(a_t|s_t)$ and importance weight truncation with bias correction to reduce variance for off-policy distribution shifts. While ACER off-policy algorithm was designed to reduce variance and improve training stability, we find it also successfully corrected the distribution shift corresponding to the replay episode.Formally, the policy gradient of ACER off-policy algorithm is given by:

$$G_{off-policy} = \overline{\rho}_t \nabla_\theta log \pi_\theta(a_t|x_t)[Q^{ret}(x_t, a_t) - V_\theta(x_t)] + E(a \sim \pi)\left([\frac{\rho_t(a) - c}{\rho_t(a)}]_+ \nabla_\theta log \pi_\theta(a|x_t)[Q_\theta(x_t, a) - V_\theta(x_t)]\right) \quad (4)$$

where $\overline{\rho}_t$ is the truncated importance weight, $\overline{\rho}_t = min(c, \rho_t)$ with $p_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$, $[x]_+ = x$ if $x > 0$ and it is zero otherwise(with c constants). On the basis of ACER, we have added two losses [14], namely $L_{policy-loss}$ and $L_{value-loss}$. These two losses use KL divergence and L2 norm respectively to fit the differences between the target policy and the behavioral policy. The pseudocode of on-policy flow is shown in algorithm 2. The agent selects a episode from memory, and uses this episode to update network parameters through ACER's off-policy algorithm.

E. Overall process

In this approach, a set of targets is first defined, and each target is iterated over. During each target iteration, policy updates are performed by invoking the on-policy learning (Algorithm [1]). In the process of policy updating, according

Algorithm 2 OPR:off-policy

Reset gradients $d\theta \leftarrow 0$
Initialize parameters $\theta^{'} \leftarrow \theta$
get an episode from memory $\{(s_t, a_t, r_t, \pi_t, Q_t)|t \in (1, \ldots, k)\}$.
for $i \in (1, ..., k)$ **do**
  calculate Retrace $Q(s_t, a_t)^{ret} \leftarrow r_t + \gamma Q^{ret}$
  update network parameter $\theta \leftarrow \theta + \alpha G_{off-policy}$
end for

to the pre-defined playback ratio r, a random number k is generated using Poisson distribution, which is used to determine the number of iterations of the off-policy learning algorithm (Algorithm [2]). Through multiple off-policy optimizations, past experience can be more effectively utilized. Next, at the end of each target iteration, we collect a certain amount of replay episode and store it in memory. Specifically, by performing a certain number of environment interactions, we collect the state s_t, action a_t, reward r_t, policy π_t and Q-value estimate Q_t for each round are stored stand up, the memory capacity of each target is $\frac{m}{n}$ episodes. After collecting the optimal trajectory, agent continues to learn the next navigation target. Through this algorithm, we can learn the learned policy while learning the new target policy. This approach takes full advantage of both online and offline policy optimization to improve the efficiency and performance of the algorithms. The pseudocode of complete algorithm flow is shown in algorithm 3.

Algorithm 3 OPR

//Assume ratio of replay r.
Define target $\in (1, ..., n)$.
for $i \in (1, \ldots, n)$ **do**
  while The maximum number of episodes not reach **do**
    Call OPR on-policy, Algorithm[1]
    k$\leftarrow Possion(r)$
    for $j \in (1, ..., k)$ **do**
      Call OPR off-policy, Algorithm[2]
    end for
  end while
  for $j \in (1, \ldots, \frac{m}{n})$ **do**
    Collect episodes $(s_t, a_t, r_t, \pi_t, Q_t)$ and store them in memory
  end for
end for

IV. Experiments

In our study, we trained and evaluated our approach in the AI2-THOR environment [7], which consists of 10 navigation targets, namely ['AlarmClock', 'Book', 'Bowl', 'CoffeeMachine', 'Kettle', 'Plate', 'Pan', 'Toaster', 'Pot', 'Fridge']. We used the reward function proposed in [8], where finding an object rewards the agent with 5, and taking a step results in a reward of -0.01. Our experiments focus on the plasticity and stability of the network to new navigation targets, so our experiments are carried out in the same environment, in order to avoid the interference of different environments on the navigation model

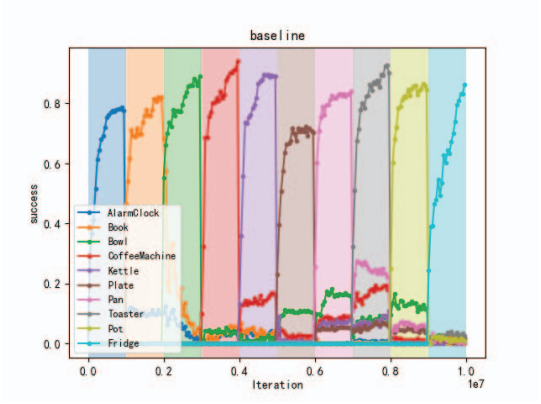

Fig. 2. The experimental results of training a agent with actor-critic structure on multiple navigation targets without using experience replay. The x-axis represents the time steps of all tasks, and the y-axis represents the success rate on each task. Our results show that this approach leads to catastrophic forgetting.

A. Baseline

we present the experimental results of a actor-critic network trained without using experience replay. The curve of the network is presented in Figure 1, which shows the performance of the network on all tasks. Each color represents the performance on a different task, and the background color indicates the task on which the network is currently being trained. We saved 10 models for each navigation target and tested them on all previously learned navigation targets. Our results show that when experience replay is not used, the model forgets the previously learned tasks.

B. OPR

Figure 2 displays the experimental results of using OPR to address catastrophic forgetting in visual navigation. The figure shows that the model trained with OPR performs significantly better than the model trained without OPR. One reason for this is that OPR uses experience replay, allowing the model to learn previously learned navigation targets while learning new ones. Additionally, OPR only saves the optimal policy in memory, avoiding the issue of forgetting caused by the standard FIFO policy or the existence of many suboptimal trajectories in memory as proposed in [18] using the Global Distribution Matching method.

C. Limited-size Memory

In reality, the storage space of memory is limited, and it is impossible to store infinitely. We trained a total of 10000000 episodes, and set up 4 different sizes of memory to observe the performance changes, namely 50k, 30k, 5k and 2k. In terms of performance, we observe that the network exhibits very good performance when the memory size is 50k and 30k.

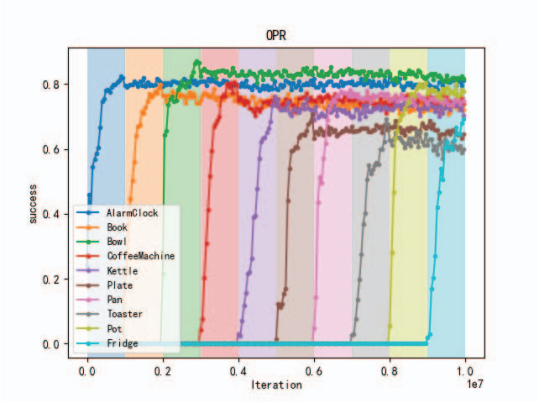

Fig. 3. The experimental results of training a agent with OPR. Our results show that OPR greatly alleviates catastrophic forgetting in visual navigation.

TABLE I
QUANTITATIVE RESULTS.WE USE THE AVERAGE SUCCESS RATE TO QUANTITATIVELY EVALUATE THE PERFORMANCE OF EACH METHOD BASED ON ALL THE ABOVE RESULTS.

	AlarmClock	Book	Bowl	CoffeeMachine	Kettle	Plate	Pan	Toaster	Pot	Fridge
OPR 50k	0.826	0.800	0.803	0.798	0.770	0.760	0.751	0.743	0.737	0.728
OPR 30K	0.825	0.793	0.774	0.739	0.711	0.667	0.679	0.655	0.671	0.660
OPR 5k	0.820	0.792	0.759	0.758	0.680	0.677	0.677	0.661	0.655	0.650
OPR 2k	0.800	0.777	0.705	0.697	0.639	0.622	0.627	0.621	0.617	0.592
Baseline	0.823	0.458	0.309	0.263	0.230	0.172	0.243	0.196	0.181	0.117
EWC [13]	0.825	0.475	0.519	0.324	0.125	0.164	0.08	0.06	0.07	0.07
GDM [18]	0.930	0.218	0.206	0.185	0.175	0.094	0.083	0.051	0.037	0.007

This suggests that a larger memory capacity helps the network better capture and utilize past experience. In contrast, we observe relatively poor performance of the network when the memory size is 2k. This suggests that a small memory capacity limits the network's effective use of past experience, leading to a decrease in its performance in learning tasks. Unlike the method in [14] where half of the experience is stored in memory, the number of trajectories stored in our memory is only a small fraction of the total number of trajectories. Our approach effectively prevents forgetting and preserves the ability to solve past tasks even with a smaller memory module.

D. Quantitative Analysis

In order to compare the performance of quantitative analytical methods and effectively capture the overall performance during continual learning (including the impact of catastrophic forgetting), we propose an evaluation metric called AS (average success),$AS = \frac{1}{n}\sum_i^n S_i$, where n is the number of currently learned navigation targets, and i is the success rate of the i-th navigation target. The values in the figure represent the test results of the agent on all learned targets after learning the current navigation target. Based on the results, while the average success rate of OPR decreases as the navigation target increases, it still exhibits a favorable balance between plasticity and stability when compared to other methods.

E. Comparison to EWC and GDM

We compared our method with EWC [13] and GDM(Global Distribution Matching) [18]. We implemented and tested their approaches on our task. Firstly, GDM is also a replay-based

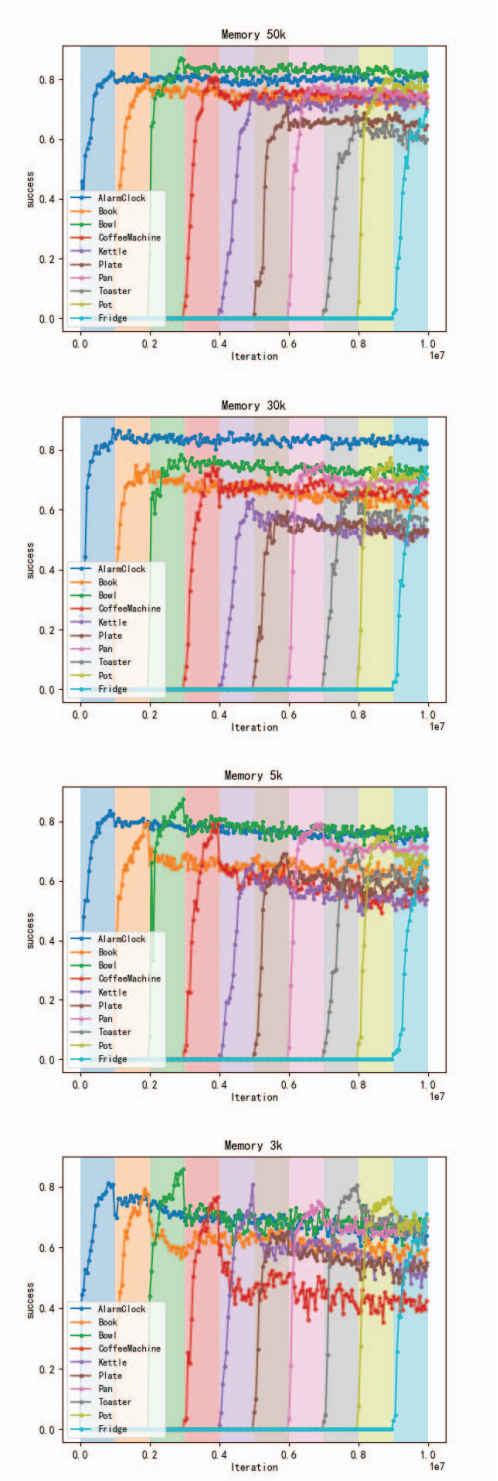

Fig. 4. Compare the network performance of different memory sizes, and test the performance of memory 50k, 30k, 5k and 3k respectively. We find that the performance of the network further improves as the memory size increases. Larger memories allow the network to better capture past experience, resulting in more stable and reliable performance on long-term tasks.

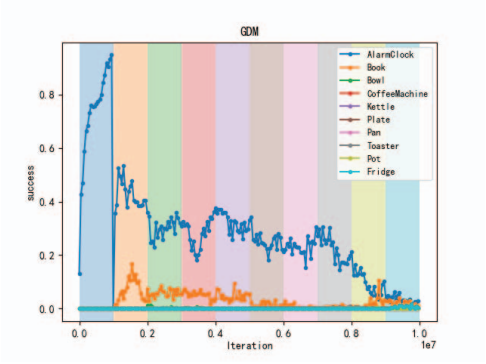

Fig. 5. The result of Global Distribution Matching

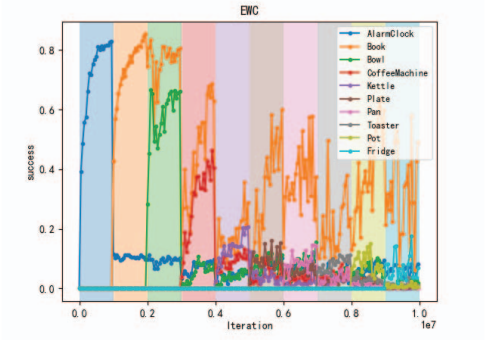

Fig. 6. The result of EWC

method that uniformly stores experiences in memory during training. This leads to a large number of suboptimal trajectory samples being stored in memory, significantly affecting the model's plasticity. Additionally, as new navigation targets are added to the memory, the proportion of excellent trajectories in the already poor-quality memory decreases, resulting in reduced model stability. On the other hand, EWC employs regularization to penalize modifications to the gradients of previously learned navigation targets. Since EWC is entirely on-policy learning, it exhibits greater plasticity compared to GDM. However, similar to GDM, EWC also suffers from a phenomenon where failure to learn one target leads to poor learning of the remaining targets. As both EWC and GDM utilize previously learned knowledge, incorrect knowledge accumulates and negatively impacts subsequent learning.

V. Conclusion

In this paper, we address the challenge of target-driven visual navigation by introducing a new paradigm called target incremental visual navigation. We propose a framework called OPR (Optimal Policy Replay) to enable continuous learning of navigation targets without relearning all targets. OPR utilizes on-policy learning to learn the current navigation target and off-policy learning to store the optimal policy for previously learned targets in memory. Our experiments show that OPR effectively mitigates catastrophic forgetting in visual navigation.

References

[1] Blösch, Michael, et al. "Vision based MAV navigation in unknown and unstructured environments." 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010.
[2] Cummins, Mark, and Paul Newman. "Probabilistic appearance based navigation and loop closing." Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007.
[3] Dissanayake, MWM Gamini, et al. "A solution to the simultaneous localization and map building (SLAM) problem." IEEE Transactions on robotics and automation 17.3 (2001): 229-241.
[4] Hadsell, Raia, et al. "Learning long-range vision for autonomous off-road driving." Journal of Field Robotics 26.2 (2009): 120-144.
[5] Kidono, Kiyosumi, Jun Miura, and Yoshiaki Shirai. "Autonomous visual navigation of a mobile robot using a human-guided experience." Robotics and Autonomous Systems 40.2-3 (2002): 121-130.
[6] Thrun, Sebastian. "Learning metric-topological maps for indoor mobile robot navigation." Artificial Intelligence 99.1 (1998): 21-71.
[7] Zhu, Yuke, et al. "Target-driven visual navigation in indoor scenes using deep reinforcement learning." 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017.
[8] Wortsman, Mitchell, et al. "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.
[9] Du, Heming, Xin Yu, and Liang Zheng. "Learning object relation graph and tentative policy for visual navigation." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16. Springer International Publishing, 2020.
[10] Du, Heming, Xin Yu, and Liang Zheng. "VTNet: Visual transformer network for object goal navigation." arXiv preprint arXiv:2105.09447 (2021).
[11] Khandelwal, Apoorv, et al. "Simple but effective: Clip embeddings for embodied ai." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.
[12] Mayo, Bar, Tamir Hazan, and Ayellet Tal. "Visual navigation with spatial attention." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.
[13] Kirkpatrick, James, et al. "Overcoming catastrophic forgetting in neural networks." Proceedings of the national academy of sciences 114.13 (2017): 3521-3526.
[14] Rolnick, David, et al. "Experience replay for continual learning." Advances in Neural Information Processing Systems 32 (2019).
[15] Atkinson, Craig, et al. "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting." Neurocomputing 428 (2021): 291-307.
[16] Kessler, Samuel, et al. "UNCLEAR: A straightforward method for continual reinforcement learning." Proceedings of the 37th International Conference on Machine Learning. 2020.
[17] Fernando, Chrisantha, et al. "Pathnet: Evolution channels gradient descent in super neural networks." arXiv preprint arXiv:1701.08734 (2017).
[18] Isele, David, and Akansel Cosgun. "Selective experience replay for lifelong learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.
[19] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. PMLR, 2016.
[20] Wang, Ziyu, et al. "Sample efficient actor-critic with experience replay." arXiv preprint arXiv:1611.01224 (2016).
[21] Thrun, Sebastian. "Learning metric-topological maps for indoor mobile robot navigation." Artificial Intelligence 99.1 (1998): 21-71.
[22] Kim, H., et al. "Autonomous helicopter flight via reinforcement learning." Advances in neural information processing systems 16 (2003).
[23] Kohl, Nate, and Peter Stone. "Policy gradient reinforcement learning for fast quadrupedal locomotion." IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004. Vol. 3. IEEE, 2004.
[24] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533.
[25] Peters, Jan, and Stefan Schaal. "Reinforcement learning of motor skills with policy gradients." Neural networks 21.4 (2008): 682-697.
[26] Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." IEEE transactions on pattern analysis and machine intelligence 40.12 (2017): 2935-2947.