IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 17, 2020, accepted July 19, 2020, date of publication July 23, 2020, date of current version August 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011438

A Survey on Visual Navigation for Artificial Agents With Deep Reinforcement Learning

FANYU ZENG[1], (Student Member, IEEE), CHEN WANG[1], AND SHUZHI SAM GE[1,2], (Fellow, IEEE)

[1]School of Computer Science and Engineering, Center for Robotics, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]Department of Electrical Computer Engineering, National University of Singapore, Singapore 119077

Corresponding author: Shuzhi Sam Ge (samge@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1813202 and Grant 61773093, in part by the National Key Research and Development Program of China under Grant 2018YFC0831800, in part by the Research Programs of Sichuan Science and Technology Department under Grant 17ZDYF3184, and in part by the Important Science and Technology Innovation Projects in Chengdu under Grant 2018-YF08-00039-GX.

ABSTRACT Visual navigation (vNavigation) is a key and fundamental technology for artificial agents' interaction with the environment to achieve advanced behaviors. Visual navigation for artificial agents with deep reinforcement learning (DRL) is a new research hotspot in artificial intelligence and robotics that incorporates the decision making of DRL into visual navigation. Visual navigation via DRL, an end-to-end method, directly receives the high-dimensional images and generates an optimal navigation policy. In this paper, we first present an overview on reinforcement learning (RL), deep learning (DL) and deep reinforcement learning (DRL). Then, we systematically describe five main categories of visual DRL navigation: direct DRL vNavigation, hierarchical DRL vNavigation, multi-task DRL vNavigation, memory-inference DRL vNavigation and vision-language DRL vNavigation. These visual DRL navigation algorithms are reviewed in detail. Finally, we discuss the challenges and some possible opportunities to visual DRL navigation for artificial agents.

INDEX TERMS Survey, visual navigation, artificial agents, deep reinforcement learning.

I. INTRODUCTION

Artificial agents refer to software or hardware entities that can perform actions in an environment independently, and include virtual robots (such as characters in games and entities in virtual environments) and real robots (such as service robots, industrial robots, and unmanned vehicles). Navigation is a key technology for artificial agents to adapt to an environment and is the precondition for other advanced behaviors.

Traditional navigation algorithms, map-based methods, include simultaneous localization and mapping (SLAM) [1] and path planning [2], [3]. SLAM research can be divided into two main categories [4]: laser SLAM and visual SLAM. Laser SLAM is a SLAM system based on a laser sensor that uses a laser to scan obstacles in the plane to build aMur-Artal 2D raster map. Although laser SLAM has achieved some success in recent years, the high price of laser sensors hinders the practical application of laser SLAM, and the efficiency of laser SLAM is susceptible to the poor weather conditions, such as rain and snow. Visual SLAM is a SLAM system based on a visual sensor (camera). A visual sensor is used to obtain environmental images, and a multi-view geometric algorithm [5] is used to construct environmental maps. Compared with laser sensors, visual sensors have the advantage of low cost and retention of semantic information about the environment. In addition, some visual SLAM algorithms have been developed by researchers. Klein and Murray [6] proposed parallel tracking and mapping (PTAM), which divides tracking and mapping into two separate tasks and processes in parallel threads. One thread addresses the task of tracking erratic hand-held motion, while the other produces a 3D map of point features from previously observed video frames. Mur-Artal *et al.* [7] proposed ORB-SLAM for expanding the versatility of PTAM to environments that are intractable for that system. Both PTAM and ORB-SLAM are based on feature extraction, but the feature method cannot process texture images well. To address this issue, Engel *et al.* [8] proposed LSD-SLAM which is a direct (feature-less) visual SLAM algorithm, and LSD-SLAM enables the construction of large-scale and consistent maps of the environment.

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

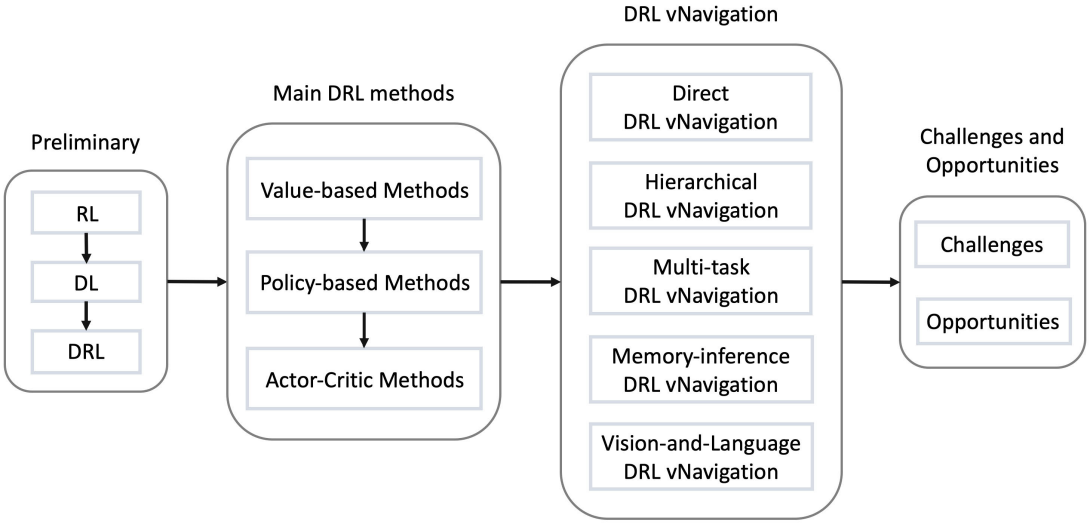


FIGURE 1. The overall architecture of this paper. We first present preliminary including RL, DL and DRL in Section II, and introduce three main DRL methods in Section II.C. Then, we review five DRL vNavigation methods in Section III. Finally, the challenges and opportunities for DRL vNavigation are discussed in Section IV.

Furthermore, Engel *et al.* [9] improved the robustness and computation speed of LSD-SLAM, and proposed direct sparse odometry (DSO). However, common disadvantages of these visual SLAM include low robustness to fast camera motion, illumination transformation, strong rotation and texture feature missing [4].

Traditional laser and visual SLAM are also model-based, and both should accurately model the environment. Nevertheless, it is difficult to model the environments effectively for some dynamic and complex scenes, which seriously affects the model-based navigation performance. In addition, the functional modularization of traditional navigation prevents their widespread applications. One prominent issue is their susceptibility to sensor noises accumulation that propagates down the pipeline from the mapping, localization to path planning, leading these algorithms with less robust performance. More importantly, they require extensive case-specific scenario-driven manual-engineering, making traditional navigation difficult to integrate with other downstream artificial intelligent tasks that have achieved superior performance with the learning methods, such as visual recognition, question answering, and other advanced intelligent tasks [10].

In recent years, the success of AlphaGo [11], [12] has promoted the rapid development of deep reinforcement learning [13]–[16]. DRL methods are end-to-end methods, in which all of the network parameters are trained jointly to avoid the accumulative error caused by the modularization of traditional navigation, and infer navigation policy directly from the visual images of the surrounding environment; they require little manual-engineering and serve as a foundation for novel AI-driven visual navigation tasks. In addition, visual navigation via DRL method is easy to integrate with downstream AI tasks to perform more advanced tasks. Therefore, an increasing number of researchers have devoted their time and effort to visual navigation with deep reinforcement learning.

Multiple surveys on visual navigation [4], [17], [18] have been published, and some surveys review a wide range of models and applications for DRL [19], [20]. Although these two types of reviews may cover parts of visual DRL navigation, the contents of visual DRL navigation are not comprehensive and systematic. In addition, novel research achievements and new challenges for visual DRL navigation have emerged. Therefore, a comprehensive review of the progress and lessons learned from state-of-the-art visual navigation algorithms based on DRL is necessary.

Thus, in this paper, we provide a comprehensive and systematic review of visual navigation based on deep reinforcement learning. A thorough investigation of the methodological evolution, issues, challenges and future potential opportunities for visual DRL navigation is discussed, which can timely facilitate practitioners and researchers in deploying, improving, and/or extending many of the current achievements in a timely manner.

The remainder of this paper is organized as follows: In Section II, we first review RL including value-based methods, policy-based methods and actor-critic methods. Then, the developments of deep learning, which contain convolutional neural networks (CNNs) and generative adversarial nets (GANs), are introduced. Finally, we review the representative DRL algorithms. Then, in Section III, different types of DRL vNavigation methods are described in detail. In Section IV, we discuss challenges existed in DRL vNavigation, and propose some possible opportunities. Finally, the conclusions are presented in Section V. The overall architecture of this paper is shown in Fig. 1.

II. DEEP REINFORCEMENT LEARNING

We consider the standard reinforcement learning setting where an artificial agent interacts with an environment over a number of discrete time steps [21]. At each time step, the agent receives a state from the environment and produces an action according to its learned policy. In return, the environment gives the agent the next state and the reward. The goal of reinforcement learning is to maximize the accumulated reward, which is a discounted sum of rewards. Fig. 2 is a schematic diagram of reinforcement learning. With the rapid development of deep learning [22], [23], DeepMind combines deep learning with reinforcement learning, and proposes deep reinforcement learning [13].

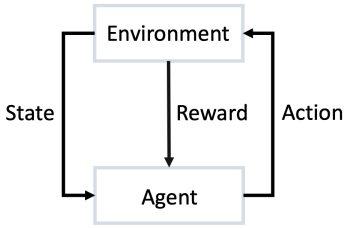


FIGURE 2. Schematic diagram of reinforcement learning. At each time step, an agent receives a state from its environment, and outputs an action. One time step later, the environment gives a next state and a reward to the agent.

In this section, we introduce the key formalism used in RL, and the three main RL algorithms which contains value function method, policy search method and actor-critic method. Next, we briefly review DL, such as CNNs and GANs. Then we introduce several main DRL algorithms: deep Q learning (DQN), deep deterministic policy gradient (DDPG), asynchronous advantage actor-critic (A3C), trust region policy optimization (TRPO) and proximal policy optimization (PPO).

A. REINFORCEMENT LEARNING

By interacting with the environment, RL agents ultimately learn the mapping relationship between an environmental state and an action, which is called policy. The reinforcement learning is a markov decision process (MDP), and the MDP can be defined as a quaternion [21]:

$$(S, A, R, P) \tag{1}$$

where S represents the state information of the environment; $s_t \in S$ is the agent's state at time t; $a_t \in A$ is the action that the agent can execute; R is the reward function that represents the reward value obtained by the agent at time t; P is the state transition probability distribution function that represents the probability of taking action a_t from state s_t to the next state s_{t+1}.

The MDP has the following characteristics:

$$P(s_{t+1}, r_{t+1}|s_1, a_1, r_1, \ldots, s_t, a_t, r_t) = P(s_{t+1}, r_{t+1}|s_t, a_t) \tag{2}$$

This equation indicates that the next state s_{t+1} is only related to the current state s_t, not to the previous states.

In reinforcement learning, artificial agents aim to maximize cumulative rewards, which can be expressed as:

$$R_t = r_1 + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3}$$

where the discount factor $\gamma \in [0, 1]$ reflects the importance of current feedback decreases over time.

The value function $V_\pi(s)$ is defined as follows:

$$V_\pi(s) = E_\pi[R_t|S_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|S_t = s] \tag{4}$$

V_π represents the expected return from state s and following π thereafter. As shown in Fig. 3, the maximum reward corresponding to $V_\pi(s)$ is the optimal value function V_π^*:

$$V_\pi^* = \max E_\pi[R_t|S_t = s] \tag{5}$$

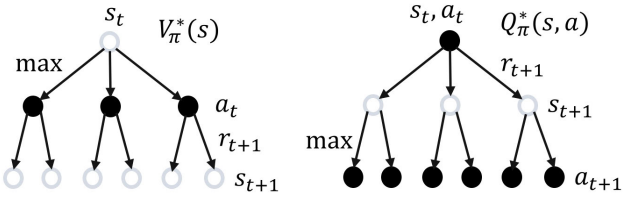


FIGURE 3. The optimal value function $V_\pi^*(s)$ corresponds to the $V_\pi(s)$ with the maximum reward, and the optimal action-value function $Q_\pi^*(s, a)$ corresponds to the $Q_\pi(s, a)$ with the maximum reward.

The action-value function $Q_\pi(s, a)$ is defined as follows:

$$\begin{aligned} Q_\pi(s, a) &= E_\pi[R_t|S_t = s, A_t = a] \\ &= E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|S_t = s, A_t = a] \end{aligned} \tag{6}$$

$Q_\pi(s, a)$ represents the expected return starting from s and taking the action a under policy π. As shown in Fig. 3, the maximum reward corresponding to $Q_\pi(s, a)$ is the optimal action-value function $Q_\pi^*(s, a)$ as follows:

$$Q_\pi^*(s, a) = \max E_\pi[R_t|S_t = s, A_t = a] \tag{7}$$

Reinforcement learning can be divided into three categories: value-based methods, policy-based methods and actor-critic methods. Actor-critic methods [24] are hybrid methods of value-based and policy-based algorithms. The classical algorithms of the value-based method and the policy-based method are Q-learning [25] and policy gradient [26], respectively.

1) VALUE-BASED METHODS

Value-based methods are to estimate the value (expected return) of being in a given state, and the optimal policy corresponds to the action with the optimal action-value function.

Q-learning is one of the most-used RL algorithms, and it updates Q through the Bellman formula:

$$Q_{i+1}(s,a) = E_\pi[R_t + \gamma \max Q_i(s_{t+1}, a_{t+1}) \mid S_t = s, A_t = a] \quad (8)$$

where Q_i converges gradually to the optimal action-value function Q_i^* when $i \rightarrow \infty$, and the optimal policy is obtained:

$$\pi^* = \arg\max_{a \in A} Q^*(s,a) \quad (9)$$

Traditional RL Q-learning is widely used and updates as follows:

$$Q(S_t, A_t) \rightarrow Q(S_t, A_t) + \alpha[r_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (10)$$

where α is the learning rate that reduces the impact of estimation errors. Unlike the Bellman formula, Q-learning does not directly assign the estimated Q value to the new Q value, but gradually approximates the target Q value. However, Q-learning is a table-based reinforcement learning in which the state space S_t and action space A_t must be finite sets. However, in the real situation, the sets of S_t or A_t are large or continuous sets, which table-based reinforcement learning cannot process. Therefore, function approximation must be used to express $Q(S_t, A_t)$, and the neural networks have good non-linear fitting characteristics. Fig. 4 shows the nonlinear fitting of neural network in which the neural networks replace the Q table.

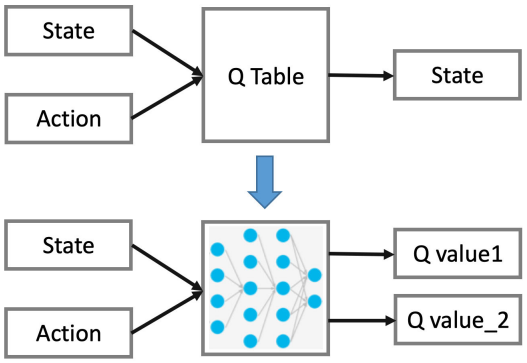


FIGURE 4. Nonlinear fitting of neural networks. Neural networks have good non-linear fitting characteristics in place of Q table.

2) POLICY-BASED METHODS

Different from the indirect policy computation of value-based methods, policy-based methods directly search for an optimal policy. Policy gradient is the most-used approach of policy-based methods, which computes an estimator of the agent's policy gradient by a stochastic gradient ascent algorithm.

Let $\pi(a|s;\theta)$ be a policy with parameters θ, which is updated by performing gradient ascent on the expectation $\mathbb{E}[R_t]$. Policy gradient algorithms adjust the policy by updating parameters θ in the direction as follows [27]:

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta log\pi(a_t|s_t;\theta)R_t] \quad (11)$$

In equation(11), $\hat{g}$ is an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$, and it is possible to reduce the variance of this estimate while keeping it unbiased. To achieve this objective, Williams [27] subtracted a learned function called baseline $b_t(s_t)$ from the return, and the resulting gradient estimator took the following form [27]:

$$\hat{g}^{PG} = \hat{\mathbb{E}}_t[\nabla_\theta log\pi(a_t|s_t;\theta)(R_t - b_t(s_t))] \quad (12)$$

where $\hat{g}^{PG}$ is also an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$.

3) ACTOR-CRITIC METHODS

Actor-critic (AC) methods are hybrid approaches that combine the advantages of policy-based and value-based methods. The AC architecture is shown in Fig. 5. The actor, which is a policy network, is to choose an action. The critic, which is a value network, is used to evaluate the advantage of the action maded by the actor.

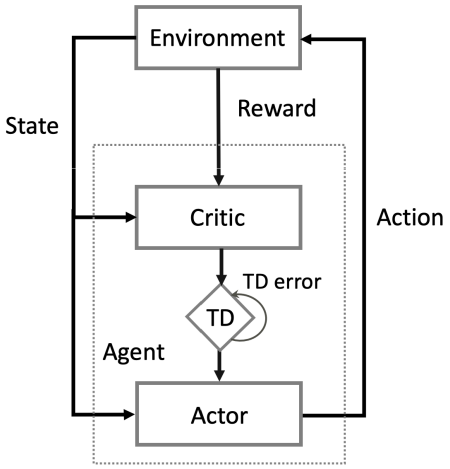


FIGURE 5. The architecture of AC. The actor is to choose an action, and the critic is to evaluate the advantage of the action chosen by the actor.

Equation(13) is called the actor-critic method in which the actor is a reference to the learned policy π and the critic refers to the baseline b_t [21], [28].

Concerning the gradient estimator in equation(12), there exists an equation $b_t(s_t) \approx V^\pi(s)$, and $R_t - b_t(s_t)$ can be seen as an estimate of the advantage of action a_t under state s_t. As the numerical value of $Q^\pi(s,a)$ equals the value of R_t, the advantage function can be rewritten as $A(a_t, s_t) = Q(s_t, s_t) - V(s_t)$. Thus, the gradient estimator in equation(12) can be rewritten as follows:

$$\begin{aligned}\hat{g}^{PG} &= \hat{\mathbb{E}}_t[\nabla_\theta log\pi(a_t|s_t;\theta)(Q(s_t,s_t) - V(s_t))] \\ &= \hat{\mathbb{E}}_t[\nabla_\theta log\pi(a_t|s_t;\theta)\hat{A}_t]\end{aligned} \quad (13)$$

B. DEEP LEARNING

In the 1880s, Hinton *et al.* [29] first introduced error backpropagation into multi-layer neural network training, which is the foundation for the later widespread usage of back propagation in deep learning. In 2006, Hinton and Salakhutdinov [30] formally proposed the concept of deep learning in which the unsupervised learning method is used to train the algorithm layer by layer and then the supervised backpropagation algorithm is used for tuning.

Lecun and Bottou [31] first trained convolutional neural networks to successfully recognize handwritten digits by error backpropagation. In 2012, Alex *et al.* entered a submission [32] that reduced the recognition error rate to 16%. The model combined several critical components that would become mainstays in deep learning models. In 2014, Simonyan and Zisserman [33] proposed a very deep convolutional network called VGGNet, which increased the depth of the neural network with more convolutional layers. VGGNet uses very small receptive fields (3×3 convolutional filter) to increase the weight layers, and more layers lead to improved performance. In the same year, Szegedy *et al.* [34] proposed GoogLeNet, which broadened the network structure and introduced an inception module. The inception module allows a network to learn input data better while further increasing the depth and width of the neural network. It can be found from the previous CNNs models that increasing the depth and width of the neural network can improve the network performance. However, simply increasing the depth will lead to vanishing or exploding gradients. To address this issue, He *et al.* [35] proposed a residual network called ResNet, which is mainly composed of residual learning blocks. The aim of the residual blocks is to solve the side effects (degradation) caused by the increasing network depth; thus, network performance can be improved with increasing network depth. Nevertheless, ResNet explicitly preserves information through additive identity transformations because many layers may contribute very little or no information. In particular, Huang *et al.* [36] proposed DenseNet to solve the vanishing of input data or gradients when they pass through the many layers to the end/beginning of a deep network. DenseNet is mainly composed of dense blocks, which create short paths from early layers to later layers. Since 2017, researchers have focused on how to design lightweight networks for resource-limited systems. Classical resource-limited CNNs architectures such as MobileNet [37] and ShuffleNet [38] are highly applicable for mobile devices.

The generative model is another important branch of deep learning. Given the training data, the goal of the generative model is to generate new samples that have a similar distribution to the training data. In 2014, Ian Goodfellow proposed generative adversarial nets [39], which have two components, including a generator G and a discriminator D. The generator G generates sample close to real sample of the input training data from random sampling noise to fool the discriminator, and the discriminator D estimates the probability that a sample came from the input training data rather than the generator. The training procedure for G is to maximize the probability of D making an incorrect discrimination. In recent years, GANs have drawn great attention, and many improved GAN methods have been developed. Radford *et al.* [40] proposed deep convolutional generative adversarial networks (DCGANs), in which CNNs are used to replace the multilayer perceptron in the original GAN. Mirza and Osindero [41] solved the problem of excessive free training in large or high pixel images by adding some conditional constraints to G and D in GAN, and this GAN is called conditional GAN (CGAN). Chen *et al.* [42] utilized mutual information to propose InfoGAN, which makes the process of generation and discrimination more controllable, and the generated results can be more easily interpreted. Zhu *et al.* [43] proposed CycleGAN based on the idea of cyclic consistency, which learns to translate an image from a source domain to a target domain in the absence of paired examples. Che *et al.* [44] proposed the wasserstein GAN (WGAN), which changes distance metrics into earth-mover (EM) distances to avoid training instability and sensitivity to hyperparameter of GANs. Moreover, many GAN variants have been developed in recent years, such as BiGAN [45], VAE-GAN [46], f-GAN [47], EBGAN [48], unrolled GAN [49], and other models [50]–[52].

Deep CNNs are representation-learning methods [22] with multiple levels of representation based on error backpropagation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher and more abstract level. With the composition of enough such transformations, very complex features of environments can be learned in a general-purpose learning procedure for artificial agents to make navigation policy. The applicability of traditional reinforcement learning has previously been limited to domains [13] in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. To solve the issues, reinforcement learning is integrated with deep CNNs to constitute deep reinforcement learning.

In general, GANs is one type of deep reinforcement learning algorithms. GANs can be viewed as actor-critic methods, in which the generator of GAN is equivalent to the actor of AC, the discriminator is equivalent to the critic. Consider an MDP where the actions set every pixel in an image [53], the environment randomly chooses either to show the state's image that the actor generates, or show a real image that the generator outputs. The reward from the environment would be 1 if the environment chose the real image and 0 if not. This MDP is stateless as the state's image generated by the actor does not affect future states.

C. MAIN ALGORITHMS OF DEEP REINFORCEMENT LEARNING

1) DEEP Q-NETWORK (DQN)

Mnih *et al.* [13] proposed DQN, which pioneers the combination of a deep convolution network and traditional reinforcement learning. Q-learning succeeds in directly

learning control policy from high-dimensional input and achieves results beyond the human level in a variety of Atari games. Compared with Q-learning, DQN mainly makes three improvements: using a deep convolution network to approximate the value function, using experience playback in the training process and setting up a separate target network to deal with time difference (TD) errors. Detailed improvements of the DQN are shown as follows:

(1) DQN replaces the Q table in Q-learning with a deep convolution network to approximate Q value, which solves the problem that Q-learning cannot be applied to S_t or A_t with large or continuous sets. Specifically, $Q(s, a|\theta_i)$ represents the output of the DQN network, and the update value function essentially updates the parameters of the network. As shown in Fig. 6, the network structure of DQN includes three convolution layers and two full connection layers.

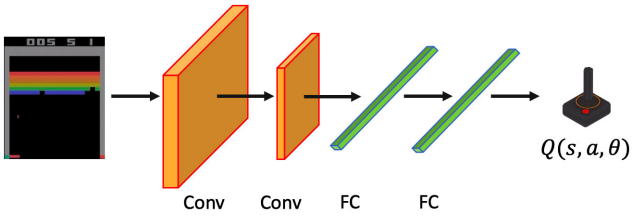


FIGURE 6. The architecture of DQN. DQN, which includes three convolution layers (Conv) and two full connection layers (FC), learns control policy directly from high-dimensional images.

(2) DQN uses experience replay [54], [55] in the training process. The agent interacts with the environment, obtains the interactive data at each time step and stores the data in an experience pool. During the training process, the agent collects the training data by uniform random sampling, and updates the network parameters by the error back-propagation algorithm. Experience replay breaks the correlation of the data, which improves the agent's convergence and stable performance.

(3) DQN sets up a separate target network to handle TD errors. The parameter update of the Q-learning network can be written as follows:

$$\theta_{t+1} = \theta_t + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta) - Q(s_t, a_t; \theta)]\nabla Q(s_t, a_t; \theta) \quad (14)$$

where $r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$ is the target Q value of TD.

To reduce the TD error, DQN sets up a separate target network whose network parameters are expressed as θ^-. The parameter update of the DQN can be written as follows:

$$\theta_{t+1} = \theta_t + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)]\nabla Q(s_t, a_t; \theta) \quad (15)$$

where the target network is set separately to keep the target Q value stable for a period of time, and decreases the correlation between the target Q value and the current Q value.

2) DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

While DQN solves problems with high-dimensional observation spaces, it can only handle low-dimensional and discrete action spaces. Lillicrap *et al.* [14] extended DQN [13] and DPG [56] to propose deep deterministic policy gradient (DDPG) algorithm in continuous domains. DDPG is based on AC and includes four neural networks: current critic network $Q(s, a|\theta^Q)$, current actor networks $\mu(s|\theta^\mu)$, target critic network $Q'(s, a|\theta^{Q'})$ and target actor network $\mu'(s|\theta^{\mu'})$, where θ^Q, θ^μ, $\theta^{Q'}$ and $\theta^{\mu'}$ are the weights of each corresponding network. The target critic network Q' and the target actor network μ' are a copy of the current critic network Q and the current actor network μ. The current critic network Q is updated by minimizing the loss function:

$$L(\theta^Q) = \mathbb{E}_{\mu'}[(y_t - Q(s_t, a_t|\theta^Q))^2] \quad (16)$$

where

$$y_t = r(s_t, a_t) + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'}) \quad (17)$$

The current actor network μ is updated by the following gradient function:

$$\nabla_{\theta^\mu}\mu \approx \mathbb{E}_{\mu'}[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_t}] \quad (18)$$

3) ASYNCHRONOUS ADVANTAGE ACTOR-CRITIC (A3C)

Mnih *et al.* [15] further proposed asynchronous advantages actor-critic (A3C). The architecture of A3C is shown in Fig. 7. The A3C algorithm uses the actor-critic framework and introduces asynchronous training and advantage function, which accelerates the training speed of the algorithm in parallel threads. The actor network outputs actions, and critic network evaluates the action selections.

A3C does not need experience replay, and its asynchronous training guarantees the diversity of exploration and exploitation. The agent in each thread interacts with the environment in parallel, which reduces the correlation of training samples and improves the agent's learning speed.

When action function $Q(a_t, s_t) \geq 0$, the policy gradient will be greater than or equal to zero and the probability of each action will increase, which results in a large policy's variance and slows down the learning speed of the agent. To address this issue, A3C introduces advantage function $A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$, and $A(a_t, s_t)$ estimates the advantage of an action a_t in state s_t. If $A(a_t, s_t) > 0$, the probability of the action a_t increases; otherwise, the probability of a_t decreases.

As entropy can measure the uncertainty of probability distribution, the policy's entropy is incorporated into A3C, and a larger entropy can prevent A3C from converging to a suboptimal policy. Therefore, the policy gradient update for A3C is defined as follows:

$$\begin{aligned} d\theta &\leftarrow d\theta + \nabla_{\theta'} log\pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v)) \\ &\quad + \beta\nabla_{\theta'} H(\pi(s_t; \theta')) \\ d\theta_v &\leftarrow d\theta_v + \partial(R - V(s_i; \theta'_v))/\partial\theta'_v \end{aligned} \quad (19)$$

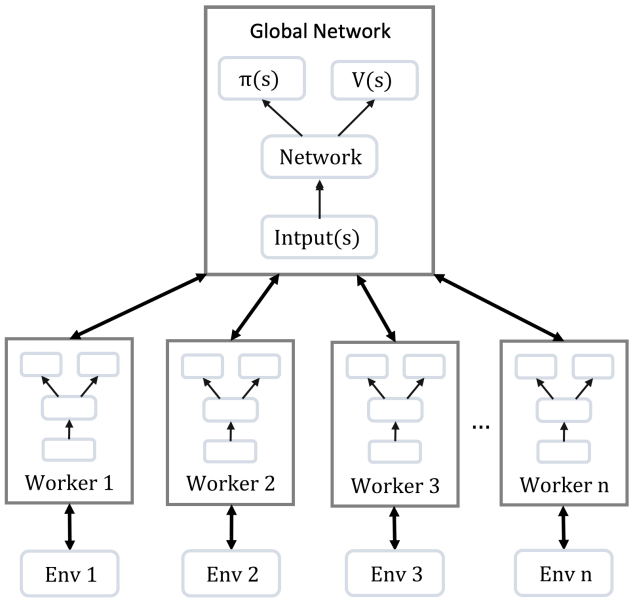


FIGURE 7. The architecture of A3C. Each AC worker is trained in parallel threads to get gradients, and the gradients are applied to global network for parameter update.

where θ and θ_v are the global parameters regarding to the action estimation and value estimation, respectively. θ' and θ'_v are the local parameters regarding to the action estimation and value estimation, respectively.

4) PROXIMAL POLICY Optimization(PPO)

When a DRL agent interacts with its environment, the state sequences of each interaction change a lot, leading to fluctuations in rewards. Therefore, DRL algorithms (such as DQN and A3C) have unstable fluctuations during training. Researchers wonder whether they can find a method to reduce such fluctuations while maintaining a steady improvement in policy. Schulman *et al.* [57] found a calculation method to measure the advantages and disadvantages of a policy and proposed the trust region policy optimization (TRPO) [57], which guarantees monotonic policy improvement after each round of parameter update. The objective function of TRPO is defined as follows:

$$\max_{\theta} \hat{\mathbb{E}}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t]$$
$$\hat{\mathbb{E}}_t[KL[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)]] \leq \delta \tag{20}$$

where θ and θ_{old} are the parameters of the new policy and old policy, respectively. δ is the hyperparameter that determines the confidence interval. $\hat{A}_t$ is an estimation of the advantage function at time t and $\hat{\mathbb{E}}_t$ is the expectation indicating the empirical average over a finite batch of samples.

In Equation(20), the constraint δ of the TRPO is difficult to determine. Generally, the approximation of δ is calculated by a second order gradient. However, the computation will be large when the variable dimension of the objective function is high. To simplify the calculation process of TRPO, Schulman *et al.* [16] proposed the proximal policy optimization (PPO) algorithm by calculating the first derivative to approximate δ. The PPO algorithm replaces the constraint in TRPO with the truncated objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{21}$$

where $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and ϵ is the superparameter.

III. VISUAL NAVIGATION OF DEEP REINFORCEMENT LEARNING

In recent years, an increasing number of researchers have applied deep reinforcement learning to the field of visual navigation for artificial agents, including navigation, positioning, mapping, and path planning. In addition to the direct application of deep reinforcement learning in visual navigation, many scholars have proposed various types of DRL vNavigation algorithms according to the characteristics of the navigation tasks. In this section, we focus on five types of DRL vNavigation that have been commonly applied in artificial agents, and the five types include direct DRL vNavigation, hierarchical DRL vNavigation, multi-task DRL vNavigation, memory-inference DRL vNavigation, and vision-language DRL vNavigation.

A. DIRECT DRL vNavigation

Some researchers directly utilize DRL algorithms in visual navigation, where an artificial agent moves in an environment to find the goal object for rewards. During the interaction process, the agent can learn how to navigate in these environments with sparse rewards; visual navigational agents in some works [58], [59] use auxiliary tasks in training. Mirowski *et al.* [58] directly applied A3C to visual navigation in a 3D environment DeepMind Lab [60]. Learning environment models as auxiliary tasks could improve RL agents [61], [62]; hence, the authors integrated auxiliary information, such as image depth and loop closure into the A3C network leading the agents' navigation to perform better. In detail, image depth information is conducive for the agent to avoid obstacles, and the closed-loop detection can be used for efficient exploration and spatial reasoning. Similar to [58], Jaderberg *et al.* [59] also incorporated A3C with different auxiliary tasks, and the tasks included pixel control and reward prediction. In particular, pixel control maximizes the change in the pixel intensity of different regions of the input, and the reward prediction is trained on reward biased sequences to remove the perceptual sparsity of the rewards. In essence, the function of auxiliary tasks is equivalent to adding environmental constraints to artificial agents, and it increases the efficiency of reward acquisition to improve the navigation performance in environments with sparse rewards. The advantage of the auxiliary task method is that special auxiliary tasks can be added for different navigation environments to enhance the agent's performance, but its obvious disadvantage is that the auxiliary task selection mainly

depends on manual experience or large numbers of parameter adjustments.

To avoid the disadvantages of the auxiliary task method, Zhu *et al.* [63] fed the target images into actor-critic neural networks in addition to the environmental observations. The model learns a policy that jointly embeds the target and the current state; thus, there is no need to retrain the navigation model for new targets. Banino *et al.* [64] set out to leverage the computational functions of grid cells to develop an A3C agent with mammalian-like navigational abilities. These grid-like representations provide an effective basis for flexible navigation in challenging novel environments.

In addition to the mentioned direct use of DRL in visual navigation, some scholars use DRL to solve a single task in navigation according to their different requirements, such as localization, mapping and path planning problems.

In terms of localization, Chaplot *et al.* [65] proposed active neural localization (ANL), and ANL utilizes the bayesian filtering localization algorithm and A3C to minimize the number of steps needed for accurate localization. However, ANL [65] bases an assumption that the transition functions in ANL are deterministic, which does not apply well to real robot. Therefore, Gottipati *et al.* [66] proposed a hierarchical likelihood estimation approach which decouples the resolution of the likelihood resolution from the distance that the robot travels, and used advantage actor-critic (A2C) to accomplish localization task for real robot.

In terms of mapping, some researchers directly apply DRL to resolve the mapping problems. Bhatti *et al.* [67] used SLAM to reconstruct the environment in a 3D map and utilized the FasterRCNN detector to obtain a semantic map. In addition, the semantic map is integrated into the DQN, which provides more abundant environmental information for the agents' decision making. Gupta *et al.* [68] proposed cognitive mapping using value iteration networks [69]. The cognitive mapping module is responsible for storing the environmental map in spatial memory, and the spatial memory improves the planning ability of artificial agents in partially observable environments. In addition to map construction, researchers try to train artificial agents to read environmental maps. For effective use of environment maps, Brunner *et al.* [70] taught the A3C agent to read maps via position cells and to search for the shortest path in unseen mazes.

In terms of path planning, Tamar *et al.* [69] proposed a differentiable path planning called value iteration networks (VIN) and embedded it into a convolutional neural network as a DRL policy network. Hence, the VIN has differentiable path planning ability. Despite its effectiveness, VIN suffers from several disadvantages, including training instability, random seed sensitivity, and other optimization problems. To address these disadvantages, Lee *et al.* [71] reconstructed the VIN as a recursive convolutional network, which demonstrates that VIN couples the recurrent convolutions with an unconventional max-pooling activation. The standard gated recurrent update equations potentially alleviate the optimization issues that plague VIN.

B. HIERARCHICAL DRL vNavigation

In many dynamic and complex environments that have high dimension state space, artificial agents via direct DRL vNavigation would face dimensional disaster and would be unable to perform effective navigation. To solve the dimensional disaster, some researchers proposed a hierarchical DRL vNavigation. Hierarchical DRL vNavigation decomposes visual navigation into subproblems and solves each of them to generate a global navigation policy based on the hierarchical RL principle [72]. We categorize hierarchical DRL vNavigation into two types: hierarchical abstract machines (HAM) DRL vNavigation and option DRL vNavigation.

1) HAM DRL vNavigation

In many scenarios, extrinsic rewards to the agent are very sparse or absent altogether. As a result, artificial agents cannot sufficiently explore the environment to learn the optimal navigation policy. The goal of HAM DRL vNavigation is to resolve this issue, and HAM DRL vNavigation learns representations of hierarchical temporal abstraction in which intrinsic rewards motivate agents to explore the environment. In essence, HAM DRL vNavigation provides a subgoal to the agent prior to making a decision and turns an environment with sparse rewards into an environment with dense rewards.

The representative HAM DRL algorithm is the hierarchical DQN (H-DQN) [73], which is based on temporal abstraction and intrinsic motivation. H-DQN sets the value function by setting subgoals on different temporal scales. The value function at the top level is used to determine the agent's decision to obtain the subobjective of the next intrinsic reward, while the value function at the bottom layer is used to determine the agent's action to meet the top-level subobjective. Fig. 8 shows the architecture of the hierarchical DQN [73].

Although H-DQN works well in scenarios with sparse rewards, it relies on manually constructing subgoals a priori for tasks and utilizes intrinsic motivation. Furthermore, approach to design proper subgoals and intrinsic motivation is not clear and nontrivial, especially for dynamic and complex environments. Therefore, Tessler *et al.* [74] applied the knowledge of learning reusable skills to solve navigation tasks in the 3D environment of Minecraft with a similar architecture to H-DQN [73], and the HAM DRL agent can selectively transfer knowledge in the form of temporal abstractions to solve a new navigation task. However, the two approaches build an open-loop policy at the meta controller that waits until the previous subtask is finished and are not able to interrupt ongoing subtasks in principle. Hence, Oh *et al.* [75] utilized a meta-controller to learn when to update the subtask, and the meta-controller aims at the problem of reward delay. The architecture can switch its subtask at any time to make an artificial agent more efficient and flexible in learning

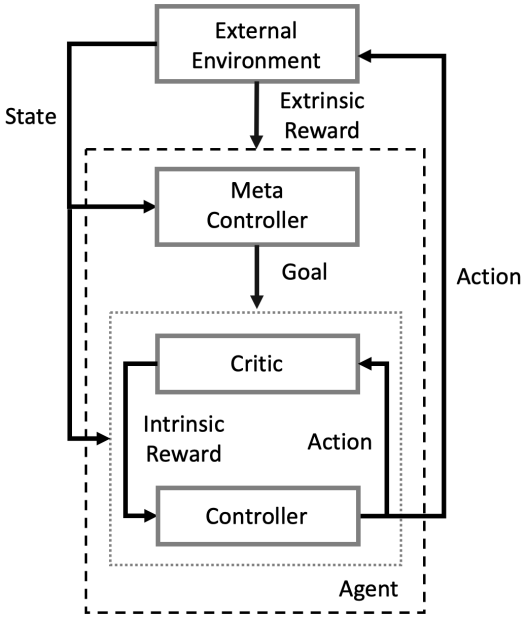


FIGURE 8. The architecture of hierarchical DQN. Both the meta-controller and controller use DQN. The meta-controller receives the states and produces a policy over goals by maximizing expected future extrinsic reward. The controller takes in the states and the current goal, and produces a policy over actions to solve the predicted goal by maximizing expected future intrinsic reward.

navigation for interrupting subtasks and for dealing with delayed sequential decision problems.

Similar to reference [74], Nachum *et al.* [76] avoided designing hand-crafted motivation or multiple tasks, and used a generic reward for the navigation agent that is specified with respect to the state space. Furthermore, Nachum *et al.* [77] developed a suboptimality concept of representation, defined in terms of the expected reward of the optimal hierarchical policy using a representation that maps observation space to goal space. This representation yields better navigation performance compared to the approach in [76].

2) OPTION DRL vNavigation

Based on option RL [78], option DRL vNavigation abstracts the navigation task into several options, which are added to the original action set as special actions. Options can be understood as a sequence of actions defined on a state subspace, and the purpose is to accomplish a navigation subgoal of following a certain policy. While options allow the representation of knowledge about courses of action that take place at different time scales, options are typically learned using subgoals and 'pseudorewards', which are provided explicitly. Hence, creating such options autonomously from data has remained challenging. To avoid hand engineering in option creation, Bacon *et al.* [79] derived stochastic policy gradient theorems for options and proposed an option-critic architecture to autonomously learn the intraoption policies and termination functions, as well as the policy over options. The option-critic architecture requires no additional rewards or subgoals to let the navigation agent find options for the expected return maximization. In addition, Tiwari [80] extend the option-critic architecture from a stochastic policy gradient to a natural gradient for learning intraoption policies and terminations, and this natural actor-critic agent also allows the autodecomposition of navigation tasks in the form of options. Compared to [79], work [81] utilizes feudal reinforcement learning to further improve the performance of the top level hierarchy called the Manager in controlling the lower level hierarchy called the Worker. This visual navigation agent outperforms the option-critic agent in the 3D environment DeepMind Lab.

These above three works [79]–[81] can learn option DRL vNavigation without giving additional reward for subgoals and can fit in learning with different foundational DRL methods. Nevertheless, these three works are learning on two time scales, which means that the low-level hierarchy should take control for a certain period. For more reactive control of low-level actors, the work in [82] proposes a hierarchical-deep deterministic policy gradient (h-DDPG) to force both levels of the hierarch learning into the same scale. Moreover, the lowest level of the hierarchy of h-DDPG can learn general basic movement skills from basic navigation tasks, and the basic movement skills that are non-task specific can transfer to different environments. However, the cost of h-DDPG includes the need to explicitly define rewards for both levels of the hierarchy. Therefore, the learning in the time scales and designing auxiliary rewards for option DRL vNavigation are contradictory, and means of having both of the advantages for artificial agents needs further study.

C. MULTI-TASK DRL vNavigation

Traditional DRL agents can navigate well in one domain but will perform poorly in other unseen domains, which means traditional DRL navigation lacks transferability. To address the transferability issue, a multitask DRL agent is used to acquire shared neural network parameters from related tasks, and the parameters represent shared navigation knowledge. Multitask DRL learning can improve data efficiency and enhance navigation transferability for artificial agents.

1) DISTILLATION DRL vNavigation

The aim of distillation DRL is to find shared knowledge across different tasks, which supports efficient transferability in complex environments. Rusu *et al.* [83] utilized policy distillation to conduct a knowledge transfer for the multitask DRL, and Fig. 9 shows the architecture of the policy distillation. In Ms. Pacman, a game maze, first learned navigation knowledge in each single domain is known as the teacher policy, and then is transferred to a multitask policy known as the student policy. In [83], the environment states of multiple tasks are assumed to share the same data distribution; hence, the convolutional filters are shared by all the tasks to retrieve the transferable features from all tasks. However, the environmental states and pixel-level inputs vary greatly in different environments. Thus, sharing the convolutional filters among

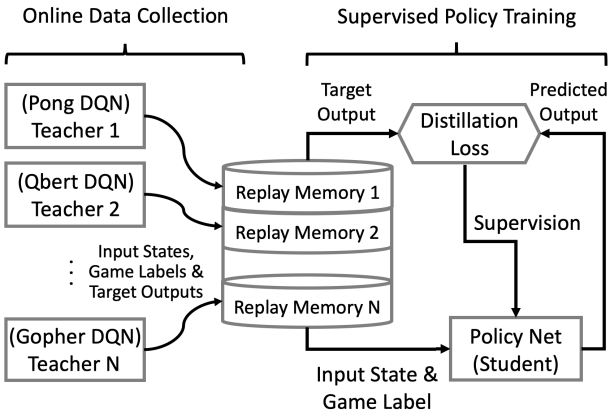


FIGURE 9. The architecture of policy distillation. The DQN agent (teacher) periodically adds gameplay to the replay memory while the policy network (student) is trained for policy distillation.

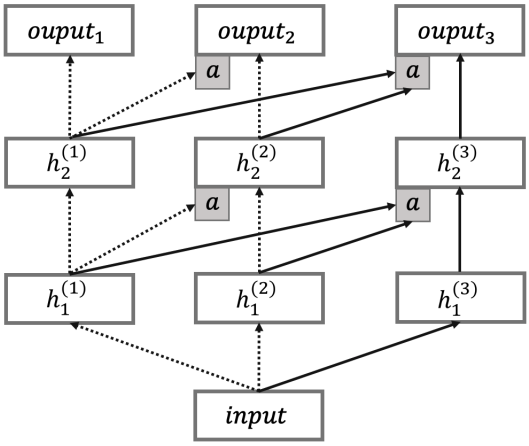


FIGURE 10. The architecture of progressive neural networks. The first two columns on the left (dashed arrows) are trained on task 1 and task 2, respectively. The grey boxes labelled a are the adapter layers. The third column has access to all previously learned features for the final task.

multiple tasks, in which the shared features contain some key task-specific features, may lead to negative transfer for some navigation tasks. To tackle the stated issue, Yin and Pan [84] proposed a new policy distillation architecture where the convolutional filters remain task-specific for each task, and a set of fully connected layers with a shared output layer are trained as the multitask policy network. Experimental results show that the latter agent [84] navigates better than the former agent [83] in Ms. Pacman. In addition, another issue is the different reward schemes between multiple tasks for distillation DRL vNavigation, which can easily lead to one navigation task dominating the learning of a shared model. To address the problem, work in [85] presented a new distilled policy that captures the shared behavior across different navigation tasks, and the new distilled policy is used to guide task-specific policies via regularization using a Kullback-Leibler (KL) divergence whose effect is akin to a shaping reward.

2) PROGRESSIVE DRL vNavigation

While distillation offers one potential solution to an agent's visual navigation, it requires a reservoir of persistent training data for all tasks, an assumption that leads to low sample efficiency and may not always hold. Progressive DRL [86] addresses the issue of distillation DRL vNavigation and can selectively leverage prior knowledge with lateral connections to previously learned features. In detail, Rusu *et al.* [86] proposed progressive neural networks, which store and extract useful navigation features through connected multiple progressive networks. When constructing a multilayer neural network to train a task, a layer of the neural network is fixed, and the feature information stored in the upper layer can be transferred to the next task. This progressive DRL vNavigation yields a more positive transfer from one environment to others. Fig. 10 shows the architecture of progressive neural networks [86].

Similar to the progressive neural network architecture [86], Mirowski *et al.* [87] proposed a progressive multicity architecture that can handle multiple tasks to obtain general navigation capability. A multicity network trains artificial agents in many cities and then freezes the neural network of specific paths and policy networks of many cities, which enables the artificial agent to acquire new knowledge while preserving prior navigation knowledge. In addition, Schaul *et al.* [88] proposed universal value function approximators (UVFAs) to construct state and target representations by decomposing multiple tasks and targets into matrices. UVFAs can generalize navigation knowledge into unseen domains with the same dynamics but different goals.

Their derivation of progressive DRL vNavigation has the advantage that performance on all considered navigation tasks is preserved but requires an ever-growing set of learned representations. In general, the navigation performance becomes more transferable when the progressive network modules increase. However, more progressive network modules mean more training parameters, which increases the training difficulty of the navigation agent. Therefore, methods to balance the number of progressive network modules and training parameters should be considered.

D. MEMORY-INFERENCE DRL vNavigation

Memory can enhance the reasoning ability of artificial agents, which is benefit to navigation performance improvement. A common internal memory is LSTM [89]/GRU [90], whose limited capacity restricts the agent's navigation ability to simple environments. Moreover, common internal memory mixes together computation and memory capacity in the network weights, and this property results in a large increase in network parameters when the memory demands of a navigation task increase. To avoid the network training difficulties caused by large numbers of parameters, some researchers have proposed external memory [13], [91]–[94].

Here, the combination of DRL and external memory is introduced to improve visual navigation performance in partially observable and large-scale environments.

Unlike internal memory, external memory structures separate computation from memory capacity, so that the number of network parameters is not tied to the memory size. Therefore, external memory has a large storage capacity while maintaining low computation. Several main external memory structures include replay buffer [13], memory networks [91], episodic memory [92], neural turing machines (NTM) [93]/ differential neural computer (DNC) [94]. As external memory has a large storage capacity, artificial agents with external memory can store more environmental information and extend their navigation ability to dynamic and complex environments.

1) REPLAY BUFFER FOR DRL vNavigation

As shown in Fig. 11, the replay buffer [13] stores the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step. In the training process, the replay buffer randomizes the data, thereby removing the correlations in the observation sequence and reducing the variance of the updates. In addition, each experience is potentially used in many weight updates, and greatly improves data efficiency.

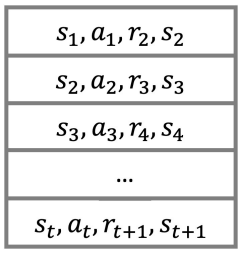


FIGURE 11. The architecture of replay buffer. Replay buffer stores agent's experience $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step.

Yin and Pan [84] proposed a new sampling framework termed hierarchical prioritized experience replay to selectively choose experiences from the replay memories of each navigation domain to perform learning on the network. The purpose of the hierarchical prioritized experience replay is to enhance the benefit of prioritization by regularizing the distribution of the sampled experiences from each navigation domain. With this prioritized replay, the overall learning for the DRL vNavigation policy is accelerated significantly. Besides taking advantage of the prioritization of replay memory in [84], some researchers store the world model in the replay buffer for navigation to reduce real-world interactions. Bruce *et al.* [95] proposed a visual robot navigation algorithm based on interactive replay, in which a rough world model is memorized from a single traversal of the environment. With the world model in interactive replay, an artificial agent interacts with the model to generate large numbers of diverse trajectories for the learning to navigate while minimizing the amount of real-world experience required by the robot. In addition, many environments are sparse-reward and often require large amounts of reward shaping. However, reward shaping limits asymptotic policy performance by preventing the policy from reaching new solutions. To address this issue, Eysenbach *et al.* [96] proposed an algorithm called search on replay buffer (SoRB), where a weighted, directed graph directly on top of the states is stored in SoRB. In the directed graph, each node corresponds to an observation and the edges between the nodes have a weight equal to their predicted distance. Using a graph search over this replay buffer, a navigation task can be automatically decomposed into a sequence of easier subgoals.

2) MEMORY NETWORKS FOR DRL vNavigation

The architecture of the memory networks is shown in Fig. 12. The update process of memory networks is as follows [91]:

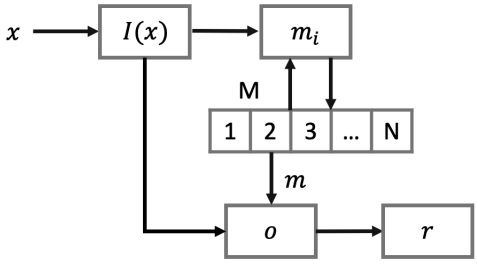


FIGURE 12. The architecture of memory networks. Memory networks consist of a memory (an array of objects indexed by m_i) and other components: input feature map I, output feature map o and response r.

(1) x is converted to an internal feature representation $I(x)$.
(2) Update memories m_i according to $m_i = G(m_i, I(x), m)$.
(3) Compute output features o according to $o = O(I(x), m)$.
(4) Decode output features o into the final output $r = R(o)$.

Memory networks can reason with inference components combined with a long-term memory component. To augment the DRL with reasoning and memory ability, Oh *et al.* [97] designed three novel DRL architectures based on memory networks: memory Q-network (MQN), recurrent memory Q-network (RMQN), and feedback recurrent memory Q-network (FRMQN). These proposed architectures store recent observations into their memory and retrieve relevant memory based on the temporal context, which leads to the DRL vNavigation agent with reasoning ability. Therefore, memory-based agents can generalize their navigation abilities to unseen or partially observable environments.

3) EPISODIC MEMORY FOR DRL vNavigation

Episodic memory [98] could provide detailed and temporally extended snapshots of the interdependency of actions and outcomes from individual experiences, and this information may be a reliable guide to decision-making precisely in situations that classical DRL algorithms cannot handle. Episodic memory may thus enable the artificial agents to [99] (1) efficiently approximate value functions over complex state spaces, (2) learn with very little data, and (3) bridge long-term dependencies between actions and rewards.

The classical architecture of episodic memory is the differentiable neural dictionary (DND) [92]. As shown in Fig. 13,

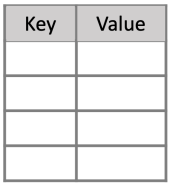


FIGURE 13. The architecture of differentiable neural dictionary. Differentiable neural dictionary has a memory module (key, value), where key and value are dynamically sized arrays of vectors, each containing the same number of vectors.

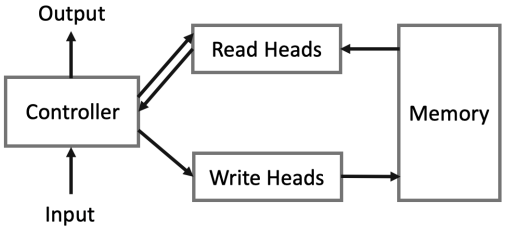


FIGURE 14. The architecture of differentiable neural computer. The controller is to receive input from external data and produce output. The read heads and write heads are used to read and write data into memory, respectively. The memory is a $N \times M$ memory matrix.

DND uses a semi-tabular representation of value, which contains the environmental state representation of the value function with slow change and the estimation of rapid update.

Some works utilize individual experiences retrieved from episodic memory to construct a cognitive map of DRL navigation. Tang *et al.* [100] integrated the cognitive mapping ability of the entorhinal cortex and the episodic memory ability of the hippocampus. Through recalling travel experiences in episodic memory, a map of the environment is built for robots to complete more cognitive navigation tasks. In work [101], episodic memory encodes the spatial temporal relationship of events to overcome the perceptual aliasing problem. And the proposed navigation model connects scenes though episodic memory retrieval to construct a sensorimotor map for robots. Moreover, some works reshape rewards by storing the navigation agent's curiosity into episodic memory to form novel rewards, which makes sparse rewards dense. Savinov *et al.* [102] computed the similarity between the current observation and the observation in episodic memory to generate new rewards. In VizDoom [103] and DeepMind Lab [60], artificial agents can quickly learn navigation capabilities, and generalize well to new environments even with very sparse rewards.

4) DNC FOR DRL vNavigation

The architecture of the differentiable neural computer is shown in Fig. 14. DNC [94] contains four modules: controller, read heads, write heads and memory. The controller is responsible for receiving input information, storing the processed data in memory, and generating ouput. The read heads read data from memory using content-based addressing or dynamic memory allocation. The read heads write data into memory using content-based addressing or temporal memory linkage. Content-based addressing enables the formation of associative data structures, dynamic memory allocation provides the write head with unused locations, and temporal memory linkages enable sequential retrieval of input sequences. In addition, the memory is a $N \times M$ memory matrix.

Some researchers have taken advantage of the large memory space of DNC, and applied DNC to visual DRL navigation. Parisotto and Salakhutdinov [104] mapped the environmental information into the DNC for a neural map, and the neural map stores the historical information of the environment map so that the learned navigation ability can be generalized to the previously unknown environments. Khan *et al.* [105] constructed an environmental value map with value iteration networks, and extended the path planning into large scale and partially observable environments. Zhang *et al.* [106] stored the internal representation of the environments into DNC, and embed the localization, motion prediction and measurement update of SLAM into the deep learning network through a soft attention mechanism. The introduction of DNC enhances the robustness and adaptability of the traditional SLAM method.

E. VISION-AND-LANGUAGE DRL vNavigation

In recent years, an increasing number of researchers have paid attention to multi-modal information [107], which provides more complete information for artificial agents' decision making [108]–[110]. Specifically, the most studied multi-modal information is vision and natural language fusion in the navigation field. Many researchers focus on the task of vision-and-language navigation (VLN) which requires artificial agents to interpret natural language instructions and to learn navigation in visual environments. Fig. 15 shows the architecture of vision-and-language navigation, in which the fusion of language instruction and vision as state inputs are fed into artificial agents for navigation policy. For example, an artificial agent is given a natural language instruction such as ''Walk forward though the door and into the living room'', and the agent should follow this language instruction to navigate from its current location to the goal position. In addition, the agent must learn to relate the language instructions with the visual information of environment. In general, most VLN methods will be evaluated on the Room-to-Room (R2R) [111] dataset which contains open vocabulary and crowd-sourced navigation instructions to guide the navigation agent to complete corresponding actions and tasks in the simulated environment.

Fried *et al.* [112] treated the VLN task as a trajectory search problem where a panoramic representation efficiently represents high-level actions and incorporates a visually grounded speaker-follower model. Wang *et al.* [113] proposed a reinforced cross-modal matching (RCM) method for VLN that enforces cross-modal grounding both locally and globally via DRL. Based on RCM, the authors introduced self-supervised imitation learning (SIL) to explore unseen environments by

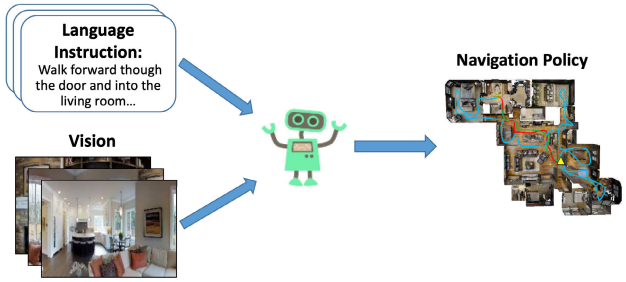


FIGURE 15. The architecture of vision-and-language navigation. Vision-language navigation is the task where an agent with visual perception navigates inside a real environment by following natural language instructions.

imitating its past and good decisions. Both approaches [112], [113] are evaluated on the R2R dataset, and they reuse pretrained vision and language modules directly in the navigation agent. To further enhance pretrained vision and language representations into domain-adapted representations. Huang *et al.* [114] defined two in-domain auxiliary tasks: cross-modal alignment (CMA) and next visual scene (NVS). With CMA and NVS, the VLN agent can learn visual and textual representations that can be transferred between different environments. The mentioned VLN algorithms assume that objects in the environment, such as offices or houses, can be formulated into instructions. Different from these VLNs, Devo *et al.* [115] focused on situations where objects in the environment cannot be specified as a navigation path, and considered 3D mazelike environments as the test bench which are very large and offer very intricate structures. This new VLN architecture can explicitly interpret the instructions and understands the direction to take along the path to navigate the environment without reference points.

IV. CURRENT CHALLENGES AND OPPORTUNITIES

In this section, we discuss the current challenges visual DRL navigation faces, and propose some research directions as opportunities that are widely open.

A. CURRENT CHALLENGES

In general, the environments are often complex, dynamic and reward-sparse. Therefore, there are two main challenges that artificial agents face: data inefficiency and poor generalization.

1) DATA INEFFICIENCY

As the inputs of the navigation agent are high-dimensional images, the navigation agent needs a large number of interactions within the environment when it learns to navigate in an environment. The interaction number will increase dramatically when the surrounding environment of the artificial agent becomes more complex and dynamic. In addition, sparse reward exacerbates the data inefficiency. The data inefficiency challenge means that the training of visual DRL navigation has poor convergence and the training time is very long.

2) POOR GENERALIZATION

Poor generalization is another issue of DRL vNavigation. There are two types of generalization cases: (1) generalization from one simulation environment to another simulation environment and (2) generalization from a simulation environment to a reality environment. In general, the reality environment is more complex and dynamic, thus the latter generalization is more difficult.

Most visual DRL navigation algorithms based on neural network architectures utilize CNNs for feature extraction and fully connected layers that map the features to a probability distribution over navigation actions. Such visual DRL navigation algorithms essentially train a reactive policy [69] for selecting actions that yield satisfactory long-term consequences in its training environment. Therefore, these DRL navigation methods suffer from poor generalization. Moreover, the data distributions in different environments vary greatly; hence, the navigation model trained in one maze is difficult to transfer to other environments.

B. OPPORTUNITIES

How to solve the two above problems is the research hotspot of visual DRL navigation. This paper lists the possible research opportunities.

1) POLICY HIERARCHY

In many dynamic and complex environments that have high-dimensional state space, artificial agents via direct DRL vNavigation would face dimensional disaster and are unable to perform effective navigation. To solve the dimensional disaster, researchers can utilize the idea of policy hierarchy [73], [76]. Policy hierarchy decomposes visual navigation into subproblems which are relatively simple tasks for artificial agents, and artificial agent solves each of them to generate a global navigation policy [74], [79] based on the hierarchical RL principle.

2) META LEARNING

In meta learning, the goal of the trained model is to quickly learn a new task from a small amount of data. Finn *et al.* [116] proposed a meta learning algorithm called model-agnostic meta learning (MAML), which trains the model's initial parameters to maximize the performance on a new task. In addition, MAML can update the parameters through one or more gradient-step computations with a small amount of data from that new task. Fig. 16 is the illustration of gradient update for policy parameters with MAML meta learning.

Meta learning is a few-shot data method, and it only needs a small amount of data for generalization. MAML [116] and its variants [117]–[119] can improve the data efficiency and transferability of visual DRL navigation for artificial agents. Therefore, incorporating visual navigation agents with the idea of meta learning is helpful to improve the DRL vNavigation performance.

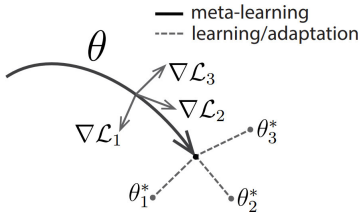


FIGURE 16. **Illustration of gradient update for policy parameters with MAML meta learning algorithm [116], which optimizes for a representation θ that can quickly adapt to new tasks.**

3) MEMORY

As humans beings, we draw on our previous experience in similar conditions from memory, when we navigate through novel environments. We can reason though the free-space, obstacles and topology of an environment with common sense rules and heuristics for navigation [68], which are based on memory.

Memory can enhance the reasoning ability [97], [101] of visual navigation model. Thus, it's helpful to store navigation experiences in memory architectures and augment artificial agents with memory functions. Memory enhances data efficiency and generalization of artificial agents in dynamic and complex environments.

4) MULTI-MODAL FUSION

Multi-modal fusion, such as speech, natural language and some other model information (such as laser radar and Inertial measurement unit), can sense sufficient environmental information [107] for artificial agents. Developing an effective multi-model fusion method enhances the data efficiency of environment, which improves the perception [120] of artificial agents to cope with dynamic and complex environments. Therefore, a visual DRL navigation agent with a multi-modal sense can learn a better policy [121]. Based on multi-modal information, an artificial agent has good adaptability to dynamic and complex environments, which helps to improve the generalization of navigation model.

V. CONCLUSION

Visual navigation is the foundational technology for artificial agents and is widely used in various fields, such as electronic games, unmanned vehicles and robotics. Visual navigation methods based on deep reinforcement learning draw much attention from researchers. This paper has provided a comprehensive and systematic review of visual DRL navigation including its developments and frontier algorithms. In addition, the current challenges and opportunities for visual DRL navigation are discussed. We hope that this survey paper will benefit researchers in the visual navigation community.

REFERENCES

[1] S. S. Ge, Q. Zhang, A. T. Abraham, and B. Rebsamen, "Simultaneous path planning and topological mapping (SP2ATM) for environment exploration and goal oriented navigation," *Robot. Auton. Syst.*, vol. 59, nos. 3–4, pp. 228–242, Mar. 2011.

[2] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Trans. Robot. Autom.*, vol. 16, no. 5, pp. 615–620, Oct. 2000.

[3] S. S. Ge, X. Lai, and A. A. Mamun, "Boundary following and globally convergent path planning using instant goals," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 240–254, Apr. 2005.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[5] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[6] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 225–234.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[8] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 834–849.

[9] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[10] X. Ye and Y. Yang, "From seeing to moving: A survey on learning for visual indoor navigation (VIN)," 2020, *arXiv:2002.11310*. [Online]. Available: http://arxiv.org/abs/2002.11310

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–16.

[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

[17] G. Huang, "Visual-inertial navigation: A concise review," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 9572–9582.

[18] X. Ye and Y. Yang, "From seeing to moving: A survey on learning for visual indoor navigation (VIN)," 2020, *arXiv:2002.11310*. [Online]. Available: http://arxiv.org/abs/2002.11310

[19] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[20] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*. [Online]. Available: http://arxiv.org/abs/1701.07274

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[23] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[24] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[25] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[26] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[27] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.
[28] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 2177–2182.
[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 696–699, 1988.
[30] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.
[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[37] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861
[38] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.
[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[40] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*. [Online]. Available: http://arxiv.org/abs/1511.06434
[41] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: http://arxiv.org/abs/1411.1784
[42] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2172–2180.
[43] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.
[44] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.
[45] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–18.
[46] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1558–1566.
[47] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: Training generative neural samplers using variational divergence minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 271–279.
[48] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–17.
[49] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–25.
[50] Y. Li, Z. Gan, Y. Shen, J. Liu, Y. Cheng, Y. Wu, L. Carin, D. Carlson, and J. Gao, "StoryGAN: A sequential conditional GAN for story visualization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6329–6338.
[51] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1429–1437.
[52] Y. Gan, K. Liu, M. Ye, and Y. Qian, "Sentence guided object color change by adversarial learning," *Neurocomputing*, vol. 377, pp. 113–121, Feb. 2020.
[53] D. Pfau and O. Vinyals, "Connecting generative adversarial networks and actor-critic methods," 2016, *arXiv:1610.01945*. [Online]. Available: http://arxiv.org/abs/1610.01945
[54] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory," *Psychol. Rev.*, vol. 102, no. 3, p. 419, 1995.
[55] J. O'Neill, B. Pleydell-Bouverie, D. Dupret, and J. Csicsvari, "Play it again: Reactivation of waking experience and memory," *Trends Neurosci.*, vol. 33, no. 5, pp. 220–229, May 2010.
[56] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1–9.
[57] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[58] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–16.
[59] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," 2016, *arXiv:1611.05397*. [Online]. Available: http://arxiv.org/abs/1611.05397
[60] C. Beattie *et al.*, "DeepMind lab," 2016, *arXiv:1612.03801*. [Online]. Available: http://arxiv.org/abs/1612.03801
[61] L.-J. Lin and T. M. Mitchell, "Memory approaches to reinforcement learning in non-Markovian domains," School Comput. Sci., Carnegie Mellon Univ., Tech. Rep. CMU-CS-92-138, 1992
[62] X. Li, L. Li, J. Gao, X. He, J. Chen, L. Deng, and J. He, "Recurrent reinforcement learning: A hybrid approach," 2015, *arXiv:1509.03044*. [Online]. Available: http://arxiv.org/abs/1509.03044
[63] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 3357–3364.
[64] A. Banino *et al.*, "Vector-based navigation using grid-like representations in artificial agents," *Nature*, vol. 557, no. 7705, pp. 429–433, May 2018.
[65] D. S. Chaplot, E. Parisotto, and R. Salakhutdinov, "Active neural localization," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.
[66] S. K. Gottipati, K. Seo, D. Bhatt, V. Mai, K. Murthy, and L. Paull, "Deep active localization," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4394–4401, Oct. 2019.
[67] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. S. Torr, "Playing doom with SLAM-augmented deep reinforcement learning," 2016, *arXiv:1612.00380*. [Online]. Available: http://arxiv.org/abs/1612.00380
[68] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2616–2625.
[69] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 2154–2162.
[70] G. Brunner, O. Richter, Y. Wang, and R. Wattenhofer, "Teaching a machine to read maps with deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[71] L. Lee, E. Parisotto, D. S. Chaplot, E. Xing, and R. Salakhutdinov, "Gated path planning networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1–12.
[72] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dyn. Syst.*, vol. 13, nos. 1–2, pp. 41–77, 2003.

[73] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.

[74] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–9.

[75] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 2661–2670.

[76] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3303–3313.

[77] O. Nachum, S. Gu, H. Lee, and S. Levine, "Near-optimal representation learning for hierarchical reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–18.

[78] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, Aug. 1999.

[79] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–9.

[80] S. Tiwari and P. S. Thomas, "Natural option critic," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 1–8.

[81] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70, Aug. 2017, pp. 3540–3549.

[82] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5174–5184, Nov. 2018.

[83] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp.'1–13.

[84] H. Yin and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.

[85] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4496–4506.

[86] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016, *arXiv:1606.04671*. [Online]. Available: http://arxiv.org/abs/1606.04671

[87] P. Mirowski, M. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, A. Zisserman, and R. Hadsell "Learning to navigate in cities without a map," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2419–2430.

[88] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.

[89] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[90] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555

[91] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2014, *arXiv:1410.3916*. [Online]. Available: http://arxiv.org/abs/1410.3916

[92] A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell, "Neural episodic control," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 2827–2836.

[93] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," 2014, *arXiv:1410.5401*. [Online]. Available: http://arxiv.org/abs/1410.5401

[94] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwinska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, p. 471, 2016.

[95] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "One-shot reinforcement learning for robot navigation with interactive replay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–10.

[96] B. Eysenbach, R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 15246–15257.

[97] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of memory, active perception, and action in minecraft," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1–22.

[98] M. Lengyel and P. Dayan, "Hippocampal contributions to control: The third way," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 889–896.

[99] S. J. Gershman and N. D. Daw, "Reinforcement learning and episodic memory in humans and animals: An integrative framework," *Annu. Rev. Psychol.*, vol. 68, no. 1, pp. 101–128, Jan. 2017.

[100] H. Tang, R. Yan, and K. C. Tan, "Cognitive navigation by neuro-inspired localization, mapping, and episodic memory," *IEEE Trans. Cognit. Develop. Syst.*, vol. 10, no. 3, pp. 751–761, Sep. 2018.

[101] W. H. Chin, Y. Toda, N. Kubota, C. K. Loo, and M. Seera, "Episodic memory multimodal learning for robot sensorimotor map building and navigation," *IEEE Trans. Cognit. Develop. Syst.*, vol. 11, no. 2, pp. 210–220, Jun. 2019.

[102] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–20.

[103] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, "ViZDoom: A doom-based AI research platform for visual reinforcement learning," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Sep. 2016, pp. 1–8.

[104] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.

[105] A. Khan, C. Zhang, N. Atanasov, K. Karydis, V. Kumar, and D. D. Lee, "Memory augmented control networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–19.

[106] J. Zhang, L. Tai, J. Boedecker, W. Burgard, and M. Liu, "Neural SLAM: Learning to explore with external memory," 2017, *arXiv:1706.09520*. [Online]. Available: http://arxiv.org/abs/1706.09520

[107] T. Baltrusaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 423–443, Feb. 2019.

[108] X. Chen and C. L. Zitnick, "Mind's eye: A recurrent visual representation for image caption generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2422–2431.

[109] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3128–3137.

[110] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "MovieQA: Understanding stories in movies through question-answering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4631–4640.

[111] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3674–3683.

[112] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3314–3325.

[113] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6629–6638.

[114] H. Huang, V. Jain, H. Mehta, A. Ku, G. Magalhaes, J. Baldridge, and E. Ie, "Transferable representation learning in vision-and-language navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7404–7413.

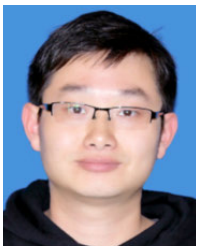
[115] A. Devo, G. Costante, and P. Valigi, "Deep reinforcement learning for instruction following visual navigation in 3D maze-like environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1175–1182, Apr. 2020.

[116] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 1126–1135.

[117] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9516–9527.

[118] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–20.

[119] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–21.
[120] S. Shiang, A. Gershman, and J. Oh, "A generalized model for multimodal perception," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 4603–4610.
[121] P. Cai, S. Wang, Y. Sun, and M. Liu, "Probabilistic end-to-end vehicle navigation in complex dynamic environments with multimodal sensor fusion," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4218–4224, Jul. 2020.



FANYU ZENG (Student Member, IEEE) received the B.S. degree from Hubei Engineering University, Xiaogan, China, in 2010, and the M.S. degree from Chongqing Normal University, Chongqing, China, in 2015. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include machine learning and computer vision.

CHEN WANG received the B.S. and M.S. degrees from Ningbo University, Ningbo, China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include reinforcement learning and computer vision.

SHUZHI SAM GE (Fellow, IEEE) received the B.Sc. degree from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 1986, and the Ph.D. degree from Imperial College London, London, U.K., in 1993. He is currently the Director with the Social Robotics Laboratory of Interactive Digital Media Institute, Singapore, and the Centre for Robotics, Chengdu, China, and a Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, on leave from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu. He has coauthored four books and over 300 international journal articles and conference papers. His current research interests include social robotics, adaptive control, intelligent systems, and artificial intelligence. He is a Fellow of the International Federation of Automatic Control, the Institution of Engineering and Technology, and the Society of Automotive Engineering. He is the Editor-in-Chief of the *International Journal of Social Robotics* (Springer). He has served/been serving as an Associate Editor for a number of flagship journals, including the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON NEURAL NETWORKS, and *Automatica*. He serves as a Book Editor for the Taylor and Francis Automation and Control Engineering Series. He served as the Vice President for Technical Activities, from 2009 to 2010, the Vice President of Membership Activities, from 2011 to 2012, and a member of the Board of Governors, from 2007 to 2009 at the IEEE Control Systems Society.