IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 22, 2020, accepted July 7, 2020, date of publication July 13, 2020, date of current version July 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3008735

Efficient Novelty Search Through Deep Reinforcement Learning

LONGXIANG SHI[1], SHIJIAN LI[1], (Member, IEEE), QIAN ZHENG[2], MIN YAO[1], AND GANG PAN[1], (Member, IEEE)

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
[2]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Corresponding author: Shijian Li (shijianli@zju.edu.cn)

ABSTRACT Novelty search, which was inspired by the nature that evolves creatures with diversity, has shown great potential in solving reinforcement learning (RL) tasks with sparse and deceptive rewards. However, most of the existing novelty search methods evolve the populations through hybrization and mutation, which is inefficient in diverging populations. In this paper, we propose a method which incorporates deep RL with novelty search to improve the efficiency of diverging the populations for novelty search. We first propose a strategy that improves the novelty of individuals generated by genetic algorithm using reinforcement learning. Based on this strategy, we propose a framework that incorporates deep RL with novelty search, and then derive an algorithm to improve the search efficiency of the novelty search for continuous control tasks. Our experimental results show that our method can improve the search efficiency of novelty search and can also provide a competitive performance compared to some of the existing novelty search methods. The implementation of our method is available at: https://github.com/shilx001/NoveltySearch_Improvement.

INDEX TERMS Reinforcement learning, novelty search, evolutionary computing, deep learning.

I. INTRODUCTION

In reinforcement learning (RL), an agent learns to find a policy in an unknown environment to maximize some notion of cumulative reward obtained from the environment [1]. Learning is especially challenging when the reward function is spar se or deceptive (i.e., the reward function contains local optima). In such cases, the agent is fragile to getting stuck in local optima and unable to properly learn if the exploration strategy fails to explore the whole environment efficiently [2], [3]. While many pioneering works have proposed to promote exploration based on state visitation frequency [3]–[5], a different approach called novelty search encourages the agent to exhibit different behaviors from the past [2], [6], [7]. Inspired by the nature that evolves the creatures with diversity, novelty search has shown great potential in solving the challenging robotic control tasks with reward functions that are sparse or deceptive [6], [8], [9].

For novelty search methods, one critical issue that determines the searching efficiency is how to efficiently evolve the populations that are different from the historical ones. Many existing works have studied this issue. One famous approach is to introduce new objectives and optimize it together with novelty, and thus yielding to multi-objective optimization. Such combination can be found in [6], [8], [10], [11]. Evolving the topologies of the policy networks when performing novelty search is an alternative way, which can also improve the efficiency of novelty search. Such method can be found in [12], [13]. However, despite the modified objective function or improved policy representation, the above two approaches still produce new individuals through hybridization and mutation, which has relatively low efficiency in diverging the populations. Moreover, as a category of black-box optimization method, evolutionary methods always have a low data efficiency, since the samples generated for novelty search are only used for evaluation of novelty or other objective functions and then discarded [14].

Several works have noticed this fact and try to reuse the samples generated by novelty search to obtain the desired populations. For instance, in [15], transfer learning is used to learn a different task with the samples generated when performing novelty search in a specific task. Cully *et al.* [9] propose a trial and error framework that uses novelty search to explore the environment and collect the samples and then train the samples with a map-based Bayesian algorithm to improve the policy. Kim *et al.* [16] propose a method that

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

reuses the samples generated by novelty search with an online adaption process that models the desired behaviors. However, existing works for utilizing the samples generated by the novelty search are mostly focused on finding some behaviors for a specific task. Using the historical data to improve the efficiency of evolving diversity populations for novelty search is absent.

In this paper, we attempt to reuse the samples in order to improve the efficiency of diverging the populations for novelty search. We analyze the distribution of behavior characteristics in one generation of novelty search, and propose a strategy that improves the novelty of individuals based on reusing the generated samples with RL. We then propose a framework that incorporates deep RL with novelty search to improve the search efficiency of novelty search. We also propose an algorithm based on the proposed framework for tasks with continuous action spaces. The proposed method is evaluated on 3 maze tasks of the well-known RL benchmark environments [17], and the results show that our method can improve the search efficiency of novelty search, and can also achieve a competitive performance comparing to some of the existing novelty search methods.

Our contribution can be summarized as below:

- We propose a strategy for evolving diversity individuals of novelty search. We show it can improve the novelty of the individuals by reusing the historical samples generated by novelty search based on off-policy RL methods.
- Based on the above strategy, we propose a framework that incorporates deep RL with novelty search to improve the search efficiency of novelty search.
- We proposed an algorithm named NS-RL to improve the search efficiency of novelty search for continuous control tasks. Our experimental results show that our method can improve the search efficiency of novelty search, and can also provide competitive performance compared to some of the existing novelty search methods.

II. RELATED WORKS

Here we summarized the related works in two categories: improving the efficiency of novelty search and improving the data efficiency of blackbox optimization methods.

A. IMPROVING THE EFFICIENCY OF NOVELTY SEARCH

As an optimization method without objective, several existing works attempt to improve the efficiency of the novelty search by introducing new objectives, and thus yielding to multi-objective optimization [18]. For example, the combination of novelty and fitness are often adopted to evolve the diversity populations [2], [6]. In [8], the authors introduce local competition and global competition along with novelty to improve the efficiency of novelty search. Quality diversity methods also involve more manually designed additional functions, or additional novelty functions and using multi-objective optimization method to conduct novelty search, which makes the novelty search even more complex [6]. Mouret and Clune [10] propose MAP-Elites algorithm, to illuminates the fitness potential of each area of the feature space. More recently, Cully and Demiris [11] propose a unifying framework of QD-optimization framework which combines multi-dimensional archive of phenotypic elites and novelty search with local competition, with a new selection method based on collection.

Evolving the topologies of the policy networks as well as the novelty together and thus improving the efficiency of novelty search is another approach. The NEAT algorithm [19] is popular in many novelty search methods, such as [12], [20] and [13]. Risi *et al.* [21] propose a method based on evolving plastic networks to improve the efficiency of novelty search.

From the perspective of evolutionary methods, the above two approaches produce new populations through hybridization and mutation, which has a relatively low efficiency in diverging the populations. Moreover, the data efficiency of the above methods are also low, since the samples generated are only used for evaluation of novelty or other objective function. Several works have proposed to address this issue. For example, in [15], transfer learning method is used to learn a different task with the samples generated when performing novelty search in a specific task. Cully *et al.* [9] propose a trial and error framework that allows robots to adapt to damage with high efficiency. The framework first uses novelty search to explore the environment and collect the samples and then train the samples with a map-based Bayesian algorithm to improve the policy. Kim *et al.* [16] propose a method that reuses the samples generated by novelty search with an online adaption process that models the desired behaviors. However, the existing works for utilizing the samples generated by novelty search are focused on finding some behaviors towards a specific task. For the novelty search method of those works still produce new individuals through hybridization and mutation, which is inefficient in diverging the populations. Using the samples generated to improve the efficiency of evolving diversity populations of novelty search have not been explored.

Comparing to previous works, in this paper we consider a different approach that reuses the generated samples by RL to diverge the populations for novelty search, which can improve the search efficiency of novelty search as well.

B. IMPROVING THE DATA EFFICIENCY OF BLACKBOX OPTIMIZATION METHODS

Several researchers have explored the combination of blackbox optimization methods with deep RL. For example, the goal exploration process-policy gradient (GEP-PG) [22] adopts a goal exploration process to fill the replay buffer and then uses DDPG [23] to learn the policies. The GEP is very close to evolution methods. Their experiments show that GEP-PG is more sample-efficient and have a low variance compared to DDPG. However, their combination does not improve the efficiency of gradient update of DDPG. Evolution guided RL (ERL) [14] introduces a hybrid algorithm that

periodically inserts the DDPG agent to the evolution optimization process, and improves the stability and efficiency in learning and exploration. In [24], the authors analyze the optimization problems with a surrogate gradient, and show that incorporating ES into surrogate gradient can improve the performance and efficiency of traditional RL methods. In CEM-RL [25], the authors combine the cross-entropy method and DDPG/TD3 [26] to accelerate the learning and improve the performance of of deep RL. However, the combination of novelty search and deep RL is absent. In this paper we explore to use deep RL method to improve the efficiency of novelty search.

III. PRELIMINARIES

In this section, we introduce some basic concepts, notations about reinforcement learning and novelty search.

A. REINFORCEMENT LEARNING

RL problems can be mathematically formulated as a Markovian Decision Process (MDP): (S, A, γ, P, R), where S is the state space, A is the action space, $\gamma \in [0, 1]$ is the discount factor, and P is the transition function that maps each state-action pair $(s, a) \in (S, A)$ to some distribution over S. In this paper we consider the standard RL setup: an agent interacting with the environment in discrete time steps; at each time step t, the agent observes a state $s \in S$, takes an action under some policy π, and receives a scalar reward $r \in R$. A policy π describes the agent's behavior, which is a probability distribution that maps a state to an action: $\pi(a|s) : S \times A \rightarrow [0, 1]$.

The return from a state s is defined as the total discounted future reward: $G_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$, where T is the terminal state. The state-action value Q is a mapping on $S \times A$ to $\mathbb{R}$, which describes the expected discounted future reward when taking action a at observation s by following policy π:

$$Q(s, a) = \mathbb{E}_{\pi}(r_1 + \gamma r_2 + \ldots + \gamma^{T-1} r_T | s_0 = s, a_0 = a)$$

The goal of RL is to find an optimal policy which maximizes the expected return from the starting state:

$$J = \max \mathbb{E}_{s_t \sim \rho}[G_t|_{s_t, a_t}]$$

Here, ρ is the state distribution under policy π.

The Bellman equation describes the recursive relationship in state-action value, and it's the fundamental principle of many RL algorithms:

$$Q(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma \mathbb{E} Q(s_{t+1}, a_{t+1})]$$

In RL, learning is off-policy if the target policy we optimize is different from the policy that interacts with the environment and generates the learning samples. Off-policy RL is attractive because it can reuse the samples that store in a buffer repeatedly, which leads to high data efficiency.

One popular off-policy RL method for dealing with continuous task is the deep deterministic policy gradient (DDPG) [23], which adopts actor-critic in policy gradient. The actor of DDPG optimizes the policy directly by using the deterministic policy gradient theorem [27]. Denoting $\pi_{\theta^\pi}(s|\theta^\pi)$ and $Q(s, a|\theta^Q)$ as the parameterized policy π and the action-value function Q respectively, the gradient of the loss function J can be calculated as:

$$\nabla_{\theta^\pi} J = \mathbb{E}_{s_t \sim \rho}[\nabla Q(s, a|\theta^Q)|_{s=s_t, a=\pi(s_t)} \cdot \nabla_{\theta^\pi} \pi(s|\theta^\pi)|_{s=s_t}] \quad (1)$$

The loss J_Q of the critic function can be calculated as:

$$J_Q = \mathbb{E}_{s_t \sim \rho}[(r_t + Q(s_{t+1}, a_{t+1})|_{\theta Q} - Q(s_t, a_t)|_{\theta Q})^2] \quad (2)$$

During learning, gradient ascent is performed on the actor function to maximize J, while gradient descent is performed on the critic function to minimize J_Q.

B. NOVELTY SEARCH

When the reward is sparse or deceptive, RL becomes especially challenging because the agent can seldom retrieve useful information from the environment. The agent is fragile to getting stuck in local optima and fails to proper learn. Novelty search is a variant of genetic algorithm that is less sensitive to the sparse and deceptive reward. It evolves the population based on how different their behaviors are from the ones that have already been evolved [15]. In novelty search, each policy π_θ is evaluated in the environment to evaluate its behavior. Based on its behavior, a domain-independent *behavior characteristic* (*BC*) is assigned to the policy. The BC for a policy is usually defined as a function that maps from its trajectory to some features of its trajectory. Denoting $trajectory(\pi) = \{< s_1, a_1, r_1, s_2 >, < s_2, a_2, r_2, s_3 > \ldots < s_{T-1}, a_{T-1}, r_{T-1}, s_T >\}$ is the trajectory generated by policy π_θ, f is a feature function, then the behavior characteristic function $bc(\pi_\theta)$ can be defined as:

$$bc(\pi_\theta) = f(trajectory(\pi_\theta)) \quad (3)$$

For example, in the 2-D maze environment, the behavior characteristics of the policy π_θ could be the final position of the agent after executing policy π_θ. The behavior characteristics of the past policies are stored in the *archive* A. The *novelty* $N(\pi_\theta, A)$ of π_θ is then calculated by measuring the average L2-norm distance of the K-nearest neighbours of $bc(\pi_\theta)$:

$$\begin{aligned} N(\pi_\theta, A) &= \frac{1}{|K|} \sum_{j \in K} \|bc(\pi_\theta) - bc(\pi_j)\|_2 \\ K &= kNN(bc(\pi_\theta), A) \\ &= \{bc(\pi_1), bc(\pi_2), \ldots, bc(\pi_j)\} \end{aligned} \quad (4)$$

The policy can be then optimized by maximizing the novelty of each generation using genetic algorithms [20] or evolution strategies [2]. In this paper we will focus on the novelty search method optimized by genetic algorithm [2].

IV. METHOD

In this section, we will introduce our method in detail. We first illustrate the data-driven novelty improvement method that

uses RL to reduce the overlap of the samples regarding the historical ones, then describe the overall framework. Finally, the practical algorithm is introduced.

A. NOVELTY IMPROVEMENT THROUGH REUSING THE HISTORICAL SAMPLES

When performing novelty search with genetic algorithm, in each generation, we first get an initial population of policies and then evaluate the novelty of each policy. The policies with a high novelty will survive and produce offspring in the next generation. According to the definition of novelty in Equation 4, policies with larger novelty denote their behavior characteristics (BCs) are far away from the historical samples, i.e., the outlier in BC space. In contrast, policies with lower novelty denote their BC are near the historical samples. If we can improve the policies with lower novelty to fill in the gap between high novelty policies and low novelty policies, then we can produce more individuals with high novelty. In addition, the BC distribution of one generation could be sparser, which can encourage the policies to explore the areas that are less sparse regarding the historical ones. Figure 1 gives an illustration of our motivation in a 2-D BC space. The blue points denote the historical BC samples in the archive, the orange points denotes the new generated BC samples by the genetic algorithm. If the improved BC (red points) are located in the middle of the outliers and the historical ones, the ''overlap'' between the new population and past policies could be reduced, and the BC distribution of one generation could be sparser.

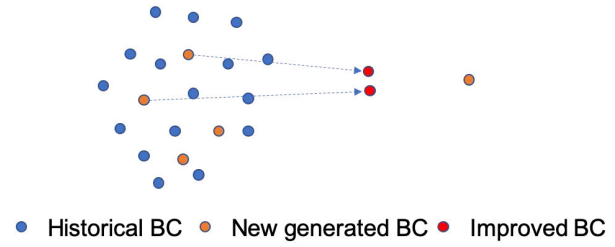


FIGURE 1. Illustration of 2-D distribution of the two dimensional behavior characteristics in one generation. The blue points denote the historical BC samples in the archive, the orange points denote the new generated BC samples by the genetic algorithm in one generation. If we can generate some new policies (red points) with BC that can fill the gap between high novelty policies and low novelty policies, the BC distribution of one generation could be sparser. Therefore we can encourage the policies to explore the areas that are less sparse regarding the historical ones.

Specifically, if the BCs of the policies are only determined by their terminal states of one episode, which is quite popular in many novelty search methods [18], we can define a distance function to measure each state of the trajectory to the target BC when executing the policy. Denoting s_i^π as the ith state observed during executing policy π, $BC(\pi^*)$ as the target BC of policy π^* and $s_T^{\pi^*}$ is the final state of policy π^*, such distance can be described as:

$$D(s_i^\pi, BC(\pi^*)) = \|s_i^\pi - s_T\| \tag{5}$$

Here s_T denotes the terminal states of execution. The above equation can be used as the per-step return for policy improvement in RL algorithms. We can then define a reward function for RL:

$$R(s_i, a_i, s_{i+1}) = -D(s_i^\pi, BC(\pi^*)) + b \tag{6}$$

Here b is a hyperparameter that denotes the distance to the target BC. Denoting π' as the policies that BCs are in the middle between historical BC and target BC, $< s_i, a_i, s_{i+1} >$ as the ith state transition pair stored in the historical trajectory, obtaining a policy π' based on a policy π_θ can be formulated as optimizing the following loss function:

$$J_\theta = \max \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}[\sum_{i=1}^{T} \gamma^T R(s_i, a_i, s_{i+1})] \tag{7}$$

We can then adopt off-policy RL methods to reuse the trajectories to improve the diversity of the population obtained from genetic algorithm. Specifically, if we use policy gradients method to optimize the policies, the hyperparameter b can be omitted because it does not contain any policy parameters:

$$\begin{aligned}\nabla J_\theta &= \nabla \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}[\sum_{i=1}^{T} \gamma^T (-D(s_i^\pi, BC(\pi^*)) + b)] \\ &= \nabla \mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}[\sum_{i=1}^{T} \gamma^T (-D(s_i^\pi, BC(\pi^*)))]\end{aligned} \tag{8}$$

In the next section we will describe a framework that uses policy gradient for instance.

B. EFFICIENT NOVELTY SEARCH FRAMEWORK

Our efficient novelty search framework combines the genetic algorithm with deep RL method, as shown in Figure 2. The deep RL agent maintains a unique actor and critic function. For each generation of the genetic algorithm, we sort the policies (also known as actors) based on their novelty, and use the BC of the top novelty policy as the target BC. All the trajectories are stored in the replay buffer. The last k policies with the lowest novelty are copied to the deep RL agent, and the RL agent then performs off-policy policy gradient using the reward calculated based on Equation 6. The deep RL agent then uses policy gradient algorithm to improve the policies towards the direction of maximum novelty, as described above.

Since the goal of the critic networks changes for each iteration, to further improve the network reusability of critic networks, we also use universal value function approximators (UVFA) [28] in the critic function. As illustrated in the framework, we introduce the goal g in the critic function. By using the UVFA, we are able to use a unique critic function to deal with the different goals and different reward functions. Through sampling m mini-batch of samples from the replay buffer, the loss of the critic function can be written as below:

$$y_i = \begin{cases} \gamma Q(s_{i+1}, \quad a_{i+1}, g_{i+1}) + R(s_i, a_i, g_i), & i+1 \neq T \\ R(s_i, a_i, g_i), & i+1 = T. \end{cases}$$

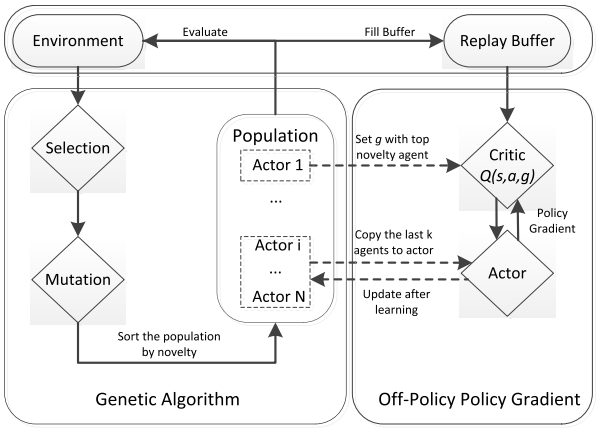


FIGURE 2. Efficient novelty search framework. Our framework incorporates deep RL with novelty search. The novelty of the populations can be improved by reusing the historical samples with deep RL.

$$J_Q = \frac{1}{m}\sum_{i=1}^{m}[y_i - Q(s_i, a_i, g_i)]^2 \tag{9}$$

During the iteration, the goal g is set to the BC of the top novelty actor of the population. After performing policy improvement with off-policy policy gradient, the actors with lower novelty in the population will be replaced with the original ones.

C. NS-RL ALGORITHM

Based on the above framework, the psudocode of the proposed algorithm is illustrated in Algorithm 1. We name it NS-RL. Before the algorithm begins, the initial population for genetic algorithm is initialized. For each iteration of the genetic algorithm, we first evaluate the novelty of each policy in one generation, store the trajectories of each episode in the experience replay, and save the BCs of the policies to the archive. The elites for each generation are guaranteed to survive in the next generation. For the last k policies with lowest novelty, we conduct off-policy policy gradient to update the parameters of them. The novelty of the improved policies are then evaluated and their trajectories are also stored in the experience replay. If the novelty of the policy is improved after learning, then we replace the corresponding policy parameters in the population. We here use DDPG [23] in the RL part for tasks with continuous action spaces (as we used in the experiments), and the tasks with discrete action spaces can be easily modified by using actor-critic methods. We also adopt the target networks for both actor and critic functions, which is popular in deep RL methods to further improve the stability of deep RL [23]. The target networks are a copy of actor and critic networks and are used for calculating the target values. Denoting θ,Ω as the parameters of actor and critic networks, θ', Ω' as the parameters of target actor and critic networks, the target networks are updated by a soft update equation:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \quad \Omega' \leftarrow \tau\Omega + (1-\tau)\Omega'$$

Algorithm 1 NS-RL Algorithm

Input: mutation function ψ, mutation rate α, population size N, number of elites E, archive $\mathcal{A}$, novelty calculate function η, policy behavior characteristic function BC, distance to the target b, number of novelty improvement policies k, experience replay $\mathcal{R}$, random generator $random() \in [0, 1)$.
Initialization:Initialize the original population pop_π with N policies, Initialize the policy network π_θ, critic network Q_Ω, actor target network $\pi_{\theta'}$, critic target network $Q_{\Omega'}$.
for $generation = 1, 2, \ldots, G$ **do**
  for $\pi \in pop_\pi$ **do**
    $trajectory = Evaluate(\pi)$.
  end for
  Store $trajectory$ in $\mathcal{R}$.
  $bc_\pi = BC(trajectory)$.
  $novelty = \eta(bc_\pi)$.
  Store bc_π in $\mathcal{A}$.
  Sort the population based on the $novelty$.
  Select the top E policies as elites, and add the N^-E policies to S.
  Select the last k policies and add in U.
  for $\pi \in U$ **do**
    Copy the parameter of π to actor: $\pi_\theta \leftarrow \pi$, $\pi'_\theta \leftarrow \pi$.
    for $learning\_steps = 1, 2, \ldots, M$ **do**
      Randomly select m mini-batch of samples (s_i, a_i, s_{i+1}) from $\mathcal{R}$.
      $r_i = -D(s_{i+1}, BC(\pi_{best})) + b$
      $g = BC(\pi_{best})$.
      $y_i = \begin{cases} r_i + \gamma Q_{\Omega'}(s_{i+1}, \pi_{\theta'}(s_{i+1}), g), i+1 \neq T \\ r_i, i+1 = T \end{cases}$.
      Perform gradient descent for Ω based on the following equation: $L = \frac{1}{m}\sum(y_i - Q_\Omega(s_i, a_i, g)^2)$
      Perform gradient ascent for θ:
      $$\nabla_\theta J = \frac{1}{m}\sum Q_\Omega(s, a, g)|_{s=s_i, a=a_i} \nabla_\theta \pi(s)|_{s=s_i}$$
      Soft update the target networks: $\theta' \leftarrow \theta + (1-\tau)\theta'$,$\Omega' \leftarrow \Omega + (1-\tau)\Omega'$.
    end for
    $trajectory = Evaluate(\pi_{\theta_\theta})$.
    $bc_{\pi_\theta} = BC(trajectory)$.
    $novelty = \eta(bc_{\pi_\theta})$.
    Store bc_{π_θ} in $\mathcal{A}$.
    if $\eta(\pi_\theta) > \eta(\pi)$ **then**
      Replace π with π_θ.
    end if
  end for
  for $\pi \in S$ **do**
    if $random() < \alpha$ **then**
      $\pi = \psi(\pi)$.
    end if
  end for
end for

Here τ is a small real number with $\tau \ll 1$. After the novelty improved step, mutation are performed on the non-elites

individuals to generate the next generation. The algorithm terminates when a certain generation of training is reached.

V. EXPERIMENT RESULTS

A. EXPERIMENTAL SETUP

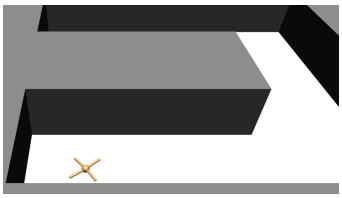
We evaluate our method in 3 locomotion-maze environments of the well-known RL benchmark environments [17], [29]. The three tasks are to control an ant-like robot to run over a specific maze with some dynamics. The action space of each task is 8-dimension with each dimension as a continuous real number from -30 to $+30$, while the observation space is 30 with continuous values. The maximum steps in one episode of the 3 environments is 500. The details of the three tasks are described below:

- **Ant-Maze**: The agent controls an ant-like robot to run over a "U"-shaped maze. The initial position of the agent is (0, 0) which locates in the lower left of the maze, while the goal position is (0, 16) which locates in the upper left of the maze.
- **Ant-Push**: The agent controls an ant-like robot to run over the maze. The initial position of the agent is (0, 0) which locates in the lower right of the maze. The target goal is (0, 19) which locates in the upper center of the maze. The red block denotes a movable obstacle. The agent must first move to the left of the block, push it to the right and then go up to reach the goal.
- **Ant-Fall**: The agent controls an ant-like robot to run over the maze. The initial position of the agent is (0, 0) which locates in the lower left of the maze. The target goal is (0, 27) which locates in the upper left of the maze. There is a chasm in the center of the maze. The agent must first push the movable block into the chasm and walk on the top of the block to cross the chasm and reach the target goal.

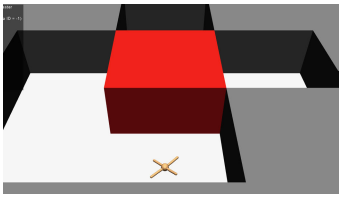
The rewards of the 3 tasks are the same: the agent receives a positive reward after it reaches the goal, else receives a zero. The rewards for the 3 maze tasks are all sparse. The conventional deep RL methods such as DQN [30] and DDPG [23] failed to learn those tasks [29]. In order to reach the goal the agent must get enough information to the maze. All the three environments are built in the MuJoCo locomotion tasks [31]. We use the OpenAI Gym[1] [32] for implementation of the environments. A 3-layer neural network is adopted to represent the policy. Each layer has 64 nodes with a ReLU activation except the output layer is activated by tanh. The critic network also uses a 3-layer neural network, each layer has 64 nodes with the hidden layer activated by ReLU. For the novelty improvement process, we perform 100 steps of gradient update using the historical samples.

We compare the NS-RL algorithm to 3 baseline methods:

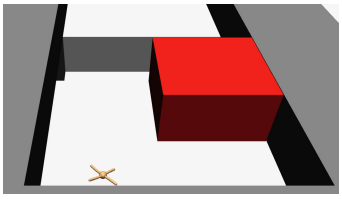
- Novelty search with genetic algorithm (NS-GA): the original novelty search method in [7]. We use genetic algorithm for optimization.



(a) Ant-Maze



(b) Ant-Push



(c) Ant-Fall

FIGURE 3. Illustration of the 3 environments used in the experiments. In each figure, the locomotion is at the start position.

- Novelty search with evolution strategies (NS-ES): this method was recently proposed in [2], which uses the evolution strategies to optimize the novelty.
- Novelty search with local competition (NS-LC): this method was proposed in [8], and uses multi-objective evolution method NSGA-II [33] to evolve diverge populations. As a quality diversity method, NS-LC has been widely used in novelty search methods [6], [13].

We conduct 3 experiments to evaluate our method. In novelty improvement evaluation, we evaluate whether the method proposed in Section IV.A could improve the novelty using the historical samples generated by novelty search. In efficiency evaluation, we evaluate the average episodes and running time to the goal to see the efficiency of our method comparing to the baseline methods. Finally, we plot the distribution of the NS-GA and NS-RL to show the evolving behavior characteristics. The implementation code of our proposed algorithm is available online.[2] The setting of hyperparameters can also be found in the code.

B. NOVELTY IMPROVEMENT EVALUATION

In this section, we evaluate the proposed strategy to show whether it can improve the novelty of the policies that have relatively low novelty in each generation. We run the NS-RL and NS-GA for 1,000 generations and measure the novelty of the last k policies before copy into deep RL agent and the novelty after learned by deep RL agent. Each method is evaluated for 5 times and we plot the average novelty

[1] http: gym.openai.com

[2] https://github.com/shilx001/NoveltySearch_Improvement

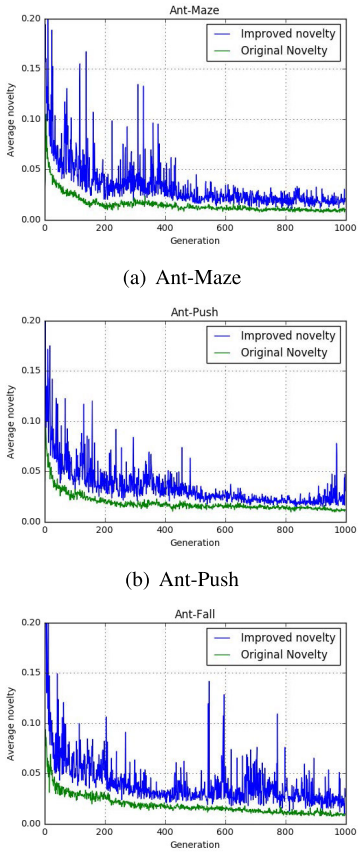


FIGURE 4. Novelty improvement evaluation of the 3 maze tasks. We evaluate the average novelty of the policies before copy into the deep RL agent and the average novelty after novelty improvement. The proposed strategy can eventually improve the novelty for the policies with lower novelty for the 3 tasks.

in Figure 4. For the 3 maze tasks, the proposed strategy can eventually improve the novelty for the policies with lower novelty.

C. EFFICIENCY EVALUATION

In this part, we compare the average episodes and running time to reach the target goal on the 3 maze environments among the 4 methods: NS-GA, NS-RL, NS-LC and NS-ES. The population size of one generation for all the 4 methods is set to 50. The novelty improvement policies for one generation of NS-RL is tuned to 10. Specifically, for the NS-ES method, we use symmetric sampling [34] to improve the learning stability and efficiency. Therefore, the number of episodes for one generation of NS-GA, NS-RL, NS-LC and NS-ES are 50,60,50,100, respectively. We run the 4 methods for 5 times and use the average value as the result. The experimental environment is: MacOS Catalina, Intel(R) Core(TM) i5(4th generation) 2.6GHz, 8G RAM, Python 3.5. The result are shown in Table 1. We also plot the distribution of the results for each method with botplot, as shown in Figure 5. In Ant-Maze and Ant-Push, NS-RL outperforms the baseline

TABLE 1. Efficiency evaluation: the upper table shows the average episode to reach the goal for the 3 tasks. The training generations to reach the goal are also shown in brackets. The lower table shows the average running time (seconds) to reach the goal for the 3 tasks.

(a) Average Episode to reach the goal

Method	Ant-Maze	Ant-Push	Ant-Fall
NS-GA	33290(665.8)	29010(580.2)	41480(829.6)
NS-LC	36190(723.8)	30110(602.2)	37860(757.2)
NS-ES	69220(692.2)	26240(262.4)	**15060(150.6)**
NS-RL	**26856(447.6)**	**21624(360.4)**	36300(605.0)

(b) Average running time (s) to reach the goal

Method	Ant-Maze	Ant-Push	Ant-Fall
NS-GA	20573.22	23062.95	41936.28
NS-LC	23632.07	24358.99	39639.42
NS-ES	42016.54	20493.44	**15225.66**
NS-RL	**19985.34**	**19930.12**	42350.00

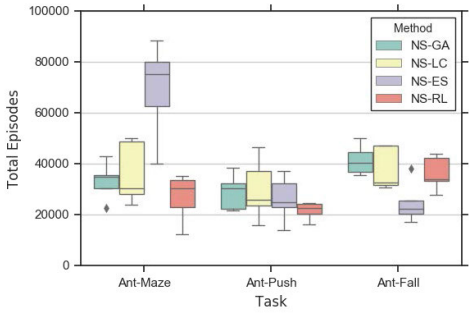
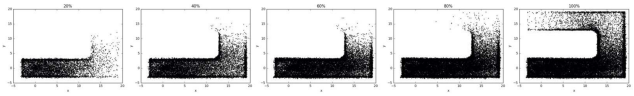


FIGURE 5. Box plot of the average episodes to reach the target goal of NS-GA, NA-LC, NS-ES and NS-RL.

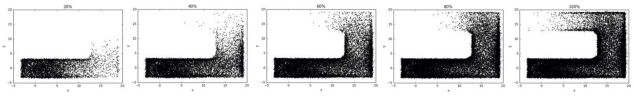
methods in both episodes and running time to reach the goal. However, in Ant-Fall task, NS-ES is better than the other 3 methods. Although our method needs more episodes in one generation to evaluate the novelty of the policies with RL improvement, our method can also outperforms the NS-GA for the 3 maze tasks. Specifically, for NS-RL, the policy improvement step contributes only 17% of the total running time of one generation. Therefore, if the hyper-parameter k is well tuned, the NS-RL could be more efficient than NS-GA. We will further discuss the experimental results by analyzing the distribution of policy BCs in the next section.

D. ANALYSIS

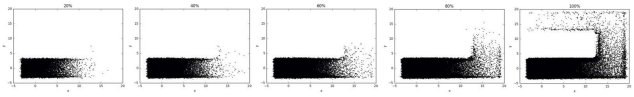
We also analyze the distribution of the policy BCs during the iteration for the 3 tasks. We analyze the median result of the 5 runs and plot the BCs of the policies for NS-GA, NS-LC, NS-ES and NS-RL. To show the evolving dynamics of the BCs, we plot 20%, 40%, 60%, 80% and 100% of the total generation that reach the target goal. Figure 6 and Figure 7 illustrate the distribution of policy BCs in Ant-Maze and Ant-Fall environment. The result for Ant-Push can also find in the Appendices. In Ant-Maze environment, NS-RL is more efficient in exploring the environment than the 3 methods. Comparing to NS-RL, the new generated population at each iteration of NS-GA is easy to trap in the crowd of historical samples, i.e, they are overlapped with historical BC. Specifically, for the 20% to 60% generation,



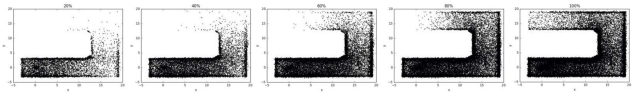
(a) NS-GA: 712 generations(35600 episodes)



(b) NS-LC: 606 generations(30300 episodes)

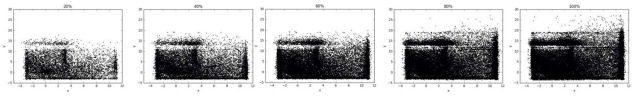


(c) NS-ES: 751 generations(75100 episodes)

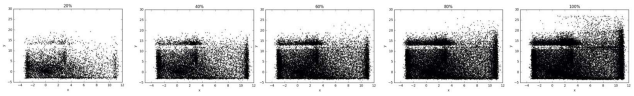


(d) NS-RL: 506 generations(30360 episodes)

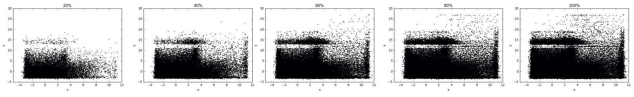
FIGURE 6. Distribution of the policy BCs for NS-GA, NS-LC, NS-ES and NS-RL in Ant-Maze task.



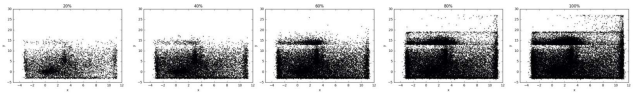
(a) NS-GA: 606 generations(30300 episodes)



(b) NS-LC: 517 generations(25850 episodes)



(c) NS-ES: 222 generation(22200 episodes)



(d) NS-RL: 563 generation(33780 episodes)

FIGURE 7. Distribution of the policy BCs for NS-GA, NS-LC, NS-ES and NS-RL in Ant-Fall task.

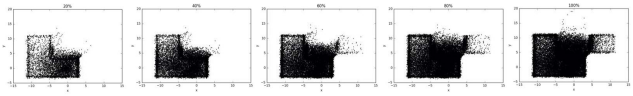
the new samples generated by NS-GA almost all fell into the lower part of the maze. In contrast, with the improvement of novelty, our method is able to explore more and avoid ''overlaps'' in the BC space than NS-GA. The result in Ant-Push is similar to Ant-Maze but less notable in observation. In the Ant-Fall environment, NS-ES outperforms the other 3 GA-based methods. As there is a chasm in the center of the maze, if the agent falls into the chasm then it will get stuck in the chasm. In this task, there are no state transitions from bottom of the chasm to the upper of the maze. Therefore, the NS-RL method performs similarly to NS-GA and NS-LC as the deep learning agent fails to go up of the barrier. We can see the edge of the chasm is more notable in Figure 7. As a consequence, our method can improve the efficiency of novelty search if the BC space has no barriers in state transition.

VI. CONCLUSION

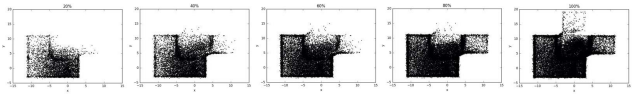
In this paper, We proposed a strategy that can improve the efficiency of novelty search by reducing the overlap of the new generated samples in each iteration. We then proposed a framework based on this strategy, which incorporates deep reinforcement learning with novelty search. An algorithm is designed based on the framework to solve the tasks with continuous action spaces. Our experimental results show that our method can improve the efficiency of novelty search, and can also provide a competitive result to some of the existing novelty search methods. The experimental results also show that if the space of policy behavior characteristics has some barriers for state transition, then the proposed method may not work. Another drawback of the proposed method is that our method is only capable for the novelty search methods with the behavior characteristic functions are only determined by its final state of execution. In the future, we will further study how to improve our method to overcome those problems.

APPENDIX A
BEHAVIOR CHARACTERISTICS OF THE ANT-PUSH TASK

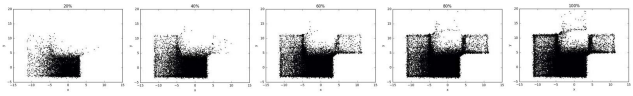
See Figure 8.



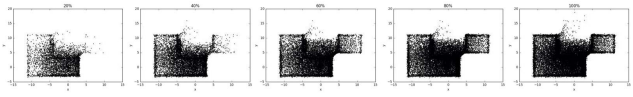
(a) NS-GA: 737 generations(36850 episodes)



(b) NS-LC: 653 generations(32650 episodes)



(c) NS-ES: 248 generations(24800 episodes)



(d) NS-RL: 379 generations(22740 episodes)

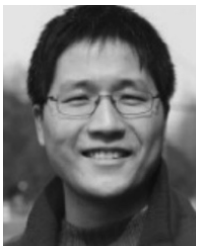
FIGURE 8. Distribution of the policy BCs for NS-GA, NS-LC, NS-ES and NS-RL in Ant-Fall.

REFERENCES

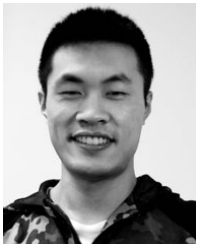
[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2011.
[2] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. Stanley, and J. Clune, ''Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5027–5038.
[3] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, ''# Exploration: A study of count-based exploration for deep reinforcement learning,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2753–2762.
[4] J. Schmidhuber, ''Formal theory of creativity, fun, and intrinsic motivation (1990–2010),'' *IEEE Trans. Auto. Mental Develop.*, vol. 2, no. 3, pp. 230–247, Sep. 2010.
[5] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, ''Vime: Variational information maximizing exploration,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1109–1117.

[6] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers Robot. AI*, vol. 3, p. 40, Jul. 2016.
[7] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," 2017, *arXiv:1712.06567*. [Online]. Available: http://arxiv.org/abs/1712.06567
[8] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proc. 13th Annu. Conf. Genetic Evol. Comput. GECCO*, 2011, pp. 211–218.
[9] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, May 2015.
[10] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," 2015, *arXiv:1504.04909*. [Online]. Available: http://arxiv.org/abs/1504.04909
[11] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 245–259, Apr. 2018.
[12] J. Gomes, P. Mariano, and A. L. Christensen, "Devising effective novelty search algorithms: A comprehensive empirical study," in *Proc. Genetic Evol. Comput. Conf. GECCO*, 2015, pp. 943–950.
[13] J. Lehman, K. O. Stanley, and R. Miikkulainen, "Effective diversity maintenance in deceptive domains," in *Proc. 15th Annu. Conf. Genetic Evol. Comput. Conf. GECCO*, 2013, pp. 215–222.
[14] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1188–1200.
[15] R. Velez and J. Clune, "Novelty search creates robots with general skills for exploration," in *Proc. Conf. Genetic Evol. Comput. GECCO*, 2014, pp. 737–744.
[16] S. Kim, A. Coninx, and S. Doncieux, "From exploration to control: Learning object manipulation skills through novelty search and local adaptation," 2019, *arXiv:1901.00811*. [Online]. Available: http://arxiv.org/abs/1901.00811
[17] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1329–1338.
[18] S. Kistemaker and S. Whiteson, "Critical factors in the performance of novelty search," in *Proc. 13th Annu. Conf. Genetic Evol. Comput. GECCO*, 2011, pp. 965–972.
[19] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002.
[20] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, Jun. 2011.
[21] S. Risi, C. E. Hughes, and K. O. Stanley, "Evolving plastic neural networks with novelty search," *Adapt. Behav.*, vol. 18, no. 6, pp. 470–491, Dec. 2010.
[22] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1039–1048.
[23] 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," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971
[24] N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein, "Guided evolutionary strategies: Augmenting random search with surrogate gradients," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 4264–4273.
[25] A. Pourchot and O. Sigaud, "CEM-RL: Combining evolutionary and gradient-based methods for policy search," 2018, *arXiv:1810.01222*. [Online]. Available: http://arxiv.org/abs/1810.01222
[26] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*. [Online]. Available: http://arxiv.org/abs/1802.09477
[27] D. Silver, G. Lever, N. Heess, T. Degris, D. Wiestra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, May 2014, pp. 387–395.
[28] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.
[29] 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.
[30] 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, pp. 529–533, Feb. 2015.
[31] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.
[32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: http://arxiv.org/abs/1606.01540
[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[34] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*. [Online]. Available: http://arxiv.org/abs/1703.03864

LONGXIANG SHI received the B.Sc. degree in software engineering from Northwestern Polytechnical University, Xi'an, China, in 2012. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. In 2018, he was a Joint Ph.D. Student with the Advanced Analytics Institute, University of Technology Sydney, Sydney, NSW, Australia, under the supervision of Prof. L. Cao. His research interests include reinforcement learning, data mining, and machine learning.



SHIJIAN LI (Member, IEEE) received the Ph.D. degree from the College of Computer Science, Zhejiang University, in 2006. From 2006 to 2008, he was a Postdoctoral Fellow with the Department of Control Science and Engineering, Zhejiang University. In 2010, he was with the Institute Telecom SudParis, France, as a Visiting Scholar. He is currently a Full Professor with the College of Computer Science, Zhejiang University. He has participated in some national projects, including the National Basic Research Program of China (973 Program), the National High Technology Research and Development Program of China (863 Program), the National Natural Science Foundation of China, and so on. He has published over 70 articles on these domains. His research interests include ubiquitous computing, artificial intelligence, and human–computer interaction.



QIAN ZHENG received the B.E. and Ph.D. degrees in computer science from Zhejiang University, China, in 2011 and 2017, respectively. He visited The Hong Kong Polytechnic University, as a Research Associate, from 2015 to 2016. He is currently a Research Fellow with the ROSE Laboratory, Nanyang Technological University. He has published several papers in international journals and conference, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, CVPR, and ICCV. His current research interests include computational photography and computer vision. He has served on Program Committee of CVPR 2018 Biometrics Workshop and an Executive Area Chairs Committee (EACC) of Vision And Learning SEminar (VALSE). He is a Reviewer of the IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE ACCESS, CVPR2020, BMVC2019, and BMVC2020.

MIN YAO received the Ph.D. degree in biomedical engineering and instrument from Zhejiang University, Hangzhou, China, in 1995. He is currently a Professor with the College of Computer Science and Technology, Zhejiang University. His research interests include computational intelligence, pattern recognition, knowledge discovery, and knowledge service.

GANG PAN (Member, IEEE) received the B.S. and Ph.D. degrees from Zhejiang University, in 1998 and 2004, respectively. From 2007 to 2008, he was with the University of California, Los Angeles, as a Visiting Scholar. He is currently a Professor with the College of Computer Science and Technology and the Vice-Director with the State Key Laboratory of CAD and CG, Zhejiang University. He has coauthored more than 100 refereed articles. He holds over 25 patents granted. His research interests include artificial intelligence, pervasive computing, brain-machine interfaces, and computer vision. He received many technical awards, including the IEEE TCSC Award for Excellence, Middle Career Researcher, in 2018, the CCF-IEEE CS Young Computer Scientist Award, in 2016, the Top-Ten Achievements in Science and Technology in Chinese Universities, in 2016, the Best Paper Award of ACM UbiComp, in 2016, the 2016 BCI Research Award Nomination, and the National Science and Technology Progress Award, in 2015. He serves as an Associate Editor for the IEEE Systems Journal, *ACM Proceedings of Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, and *Chinese Journal of Electronics*.