LEARNABLE BEHAVIOR CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

The exploration problem is one of the critical challenges in reinforcement learning (RL). Recent works tried to handle this problem with population-based methods to collect samples with different behaviors using a diverse set of learned models. To control the exploration-exploitation trade-off, prior works have to design a behavior controller to dynamically determine how to use these models in the population for sampling throughout the training process, either in a rule-based or adaptive manner, which has not been thoroughly studied. In this paper, we try the first attempt to build a unified framework to handle the behavior control problem. We formulate behavior control from two aspects: 1) *the learned models*, which are learned by learners and used to generate behaviors; 2) *the sampling strategy operators*, which map from the learned models to the actor's behavior. From this perspective, the behavior control problem is directly transformed into choosing appropriate learned models and sampling strategy operators. By properly parameterizing the learned model and sampling strategy operator, the problem becomes fully learnable, which is critical to realize more efficient behavioral control, and we call this general framework **L**earnable **B**ehavioral **C**ontrol (LBC). We introduce LBC into distributed off-policy actor-critic methods and attempt to achieve behavior control via optimizing the selection of the sampling strategy operators with a MAB-based meta-controller. One of our methods achieves 10077.52% mean human normalized score (HNS) and surpassed 24 human world records within 1B training frames in the Arcade Learning Environment (ALE), which demonstrates our significantly better sample efficiency and the State-Of-the-Art (SOTA) performance. A case study demonstrates that our methods can achieve a more fine-grained exploration-exploitation trade-off by controlling the stochasticity of behaviors at different learning stages.

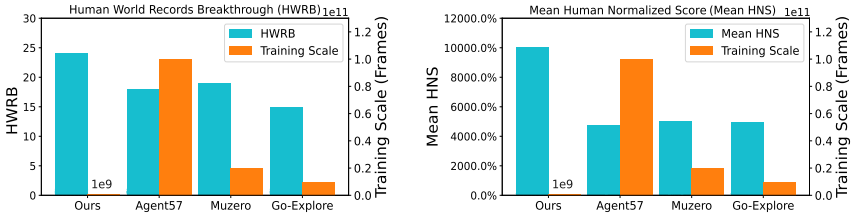

Figure 1: Performance on the 57 Atari games. Our method achieves the highest mean HNS (Bellemare et al., 2013), is the first to surpass 24 human world records (HWRB) (Toromanoff et al., 2019), and demands the least training data.

1 INTRODUCTION

Reinforcement learning (RL) has led to tremendous progress in a variety of domains ranging from video games (Mnih et al., 2015) to robotics (Schulman et al., 2015; 2017). Since RL agents only learn from experiences collected by behaviors, behavior control is crucial for building high-performing policies and balancing the exploration-exploitation trade-off (Parker-Holder et al., 2020).

Exploration is a key problem in RL, which is important for learning high-performing policies (Ecoffet et al., 2019; Badia et al., 2020b). Distributed RL (Espeholt et al., 2018; Horgan et al., 2018) have demonstrated that maintaining a population of actors to gather diverse behaviors is a promising way to boost exploration. Gathering diverse behaviors normally acquires actor-wise sampling strategy

(Sutton & Barto, 1998; Mnih et al., 2016; Espeholt et al., 2018) $e.g.$, $\epsilon - greedy$ with actor-wise ϵ. Since the behavior used to generate a trajectory is normally different from the policy on the learner, learning becomes off-policy. We call this behavior control method distributed off-policy learning (DOL). However, the sampling strategy for each actor at each training time (wall-clock) is normally deterministic and unlearned in most DOL methods (Espeholt et al., 2018; Horgan et al., 2018; Mnih et al., 2016), which struggles to adaptively balance the exploration-exploitation trade-off.

Exploration-exploitation trade-off is another critical problem in RL (Sutton & Barto, 1998). However, DOL fails to balance the exploration and exploitation trade-off adaptively with rule-based sampling strategies. This problem can be mitigated by the population-based approach, which maintains a population of models ranging from very exploratory to purely exploitative and selects a specific one to generate the behavior for each actor (Badia et al., 2020a). We call this behavior control technique the learned model selection (LMS). However, when multiple learned models overlap (See App. K) and the sampling strategy for each actor is pre-defined, the realizable behaviors for each actor will overlap, and the diversity will decrease, which is called the behavior circling problem (Parker-Holder et al., 2020).[1] To handle this problem, prior works introduced diversity-based objectives ($e.g.$, intrinsic reward (Badia et al., 2020a)) into the training loss function to boost the diversity of learned models. However, explicitly boosting diversity often hinders the optimization of already fruitful policies for rewards, and if it is misspecified fruitful policies will disregard, known as the reward-diversity trade-off problem (Parker-Holder et al., 2020). Moreover, multi-objective training requires more data to converge, caused by introducing conflicted learning objects to RL.

In conclusion, previous works have implemented behavior control differently, such as DOL and LMS, but a systematic study on achieving behavior control remains lacking. In this paper, we try the first attempt to formulate the behavior control from two aspects: the learned models to generate the behaviors, $e.g.$, the state-action value function (Badia et al., 2020a); the sampling strategy operator to map from the learned models to behaviors, $e.g.$, Boltzmann operator (Wiering, 1999). From this perspective, we detail how to construct the behavior space and select appropriate behavior. We classify behavior control techniques into two categories: i) the learned model selection, which optimizes the selection of learned models; ii) operator optimization, which optimizes the selection of the sampling strategy operators. Integrating the proposed behavior control techniques into the existing RL framework, we obtain a general framework called Learnable Behavior Control (LBC), which could provide a novel perspective of RL methods. Moreover, we formulate the behavior circling problem in LMS in Corollary 1 and demonstrate that operator optimization could be an antidote to this problem without multi-objective model training in Corollary 2. Based on that, we introduced the operator optimization into distributed off-policy actor-critic methods and obtained a family of LBC-based methods. One of these methods has achieved SOTA performance and superior learning efficiency. The main contributions of our work are:

1. **An Operator-Based View of Behavior Control.** We formulate the behavior of RL from two aspects: the learned model and the sampling strategy operator. Then, we decouple behavior control into behavior space construction and behavior selection and propose an operator-based formulation in Sec. 3.1.
2. **A General RL Framework.** Based on the behavior control formulation, we detail how existing methods achieve the behavior space construction and the behavior selection and how to improve them in Sec. 3.2 and 3.3. Integrating them into existing RL methods, we obtain a scalable RL framework LBC in Sec .3.4. We use the LBC framework to introduce a novel view of prior RL methods from the perspective of behavior control in Sec. 3.5, which leads to a better understanding of their original counterparts.
3. **A Family of LBC-Based Algorithms.** We use LBC to achieve behavior control of distributed off-policy actor-critic methods for ALE to demonstrate the effectiveness of our LBC. By choosing different behavior spaces and behavior selection methods, we can obtain a family of LBC-based methods.
4. **The State-Of-The-Art Performance.** According to Figs. 1, one of the LBC-based methods has achieved 10077.52% mean HNS and 24 unprecedented world records on ALE, outperforming the SOTA model-free algorithm Agent57. Meanwhile, our learning efficiency even outperforms the SOTA model-based method Muzero.

[1]To visualize this problem, we draw t-SNE of population-based methods with rule-based sampling strategy in Fig. 6 of App. J.3

2 PRELIMINARIES

RL can be formulated as a Markov Decision Process (Howard, 1960, MDP) defined by $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$. Considering a discounted episodic MDP, the initial state s_0 is sampled from the initial distribution $\rho_0(s) : \mathcal{S} \to \mathbb{P}(\mathcal{S})$, where we use $\mathbb{P}$ to represent the probability distribution. At each time t, the agent chooses an action $a_t \in \mathcal{A}$ according to the policy $\pi(a_t|s_t) : \mathcal{S} \to \mathbb{P}(\mathcal{A})$ at state $s_t \in \mathcal{S}$. The environment receives a_t, produces the reward $r_t \sim r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$ and transfers to the next state s_{t+1} according to the transition distribution $p(s' \mid s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{P}(\mathcal{S})$. The process continues until the agent reaches a terminal state or a maximum time step. Define the discounted state visitation distribution as $d^{\pi}_{\rho_0}(s) = (1-\gamma)\mathbb{E}_{s_0 \sim \rho_0}\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{P}(s_t = s|s_0)\right]$. The goal of reinforcement learning is to find the optimal policy π^* that maximizes the expected sum of discounted rewards, denoted by G_t (Sutton & Barto, 1998):

$$\pi^* := \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{s_t \sim d^{\pi}_{\rho_0}} \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t\right], \tag{1}$$

where $\gamma \in (0, 1)$ is the discount factor.

3 METHODOLOGY

3.1 PROBLEM FORMULATION

Learned Model Define $\Phi_{\boldsymbol{\theta},\mathbf{h}}$ as a *learned model*, *e.g.*, $Q_{\boldsymbol{\theta},\mathbf{h}}$ or $A_{\boldsymbol{\theta},\mathbf{h}}$, parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, which is trained under hyper-parameters $\mathbf{h} \in \mathbf{H}$. $\boldsymbol{\Theta}$ and $\mathbf{H}$ are the space of all possible learned model parameters and training hyper-parameters, respectively. In population-based RL, there would be a set of learned models $\{\Phi_{\boldsymbol{\theta}_1,\mathbf{h}_1}, ..., \Phi_{\boldsymbol{\theta}_{\mathrm{N}},\mathbf{h}_{\mathrm{N}}}\}$, each of which is parameterized by $\boldsymbol{\theta}_i$ and trained under its own hyper-parameters $\mathbf{h}_i$, wherein $\boldsymbol{\theta}_i \in \boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_{\mathrm{N}}\}$ and $\mathbf{h}_i \in \mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_{\mathrm{N}}\}$. M learned models can form a learned model vector $\boldsymbol{\Phi}_{\boldsymbol{\theta}_{1:\mathrm{M}},\mathbf{h}_{1:\mathrm{M}}} = (\Phi_{\boldsymbol{\theta}_1,\mathbf{h}_1}, ..., \Phi_{\boldsymbol{\theta}_{\mathrm{M}},\mathbf{h}_{\mathrm{M}}})$. Under Assumption 1, $\Phi_{\boldsymbol{\theta}_i,\mathbf{h}_i}$ can be abbreviated into $\Phi_{\mathbf{h}_i}$ and $\boldsymbol{\Phi}_{\boldsymbol{\theta}_{1:\mathrm{M}},\mathbf{h}_{1:\mathrm{M}}}$ can be abbreviated into $\boldsymbol{\Phi}_{\mathbf{h}_{1:\mathrm{M}}}$.

Assumption 1. *Assuming all learned models share the same network structure, $\mathbf{h}_i$ can uniquely index a learned model $\Phi_{\boldsymbol{\theta}_i,\mathbf{h}_i}$, which can be abbreviated to $\Phi_{\mathbf{h}_i}$.*

Sampling Strategy Operator *Sampling strategy operator* $\mathcal{F}$ is a mapping from the learned model(s) to a behavior. $\mathcal{F}$ can be classified into single-input and multi-input according to whether it is a one-to-one or many-to-one mapping. The single-input operator is a one-to-one mapping from a single model to a behavior, *e.g.*, $\mathcal{F}(\Phi_{\boldsymbol{\theta},\mathbf{h}})$, while the multi-input operator is a many-to-one mapping from multiple models to a behavior, *e.g.*, $\mathcal{F}(\Phi_{\boldsymbol{\theta}_1,\mathbf{h}_1}, ..., \Phi_{\boldsymbol{\theta}_{\mathrm{N}},\mathbf{h}_{\mathrm{N}}})$ which is a N-input operator corresponding to a N-to-one mapping. For any sampling strategy operator $\mathcal{F}_{\boldsymbol{\psi}}$ parameterized by $\boldsymbol{\psi}$, there exists a family of sampling strategy operators $\mathcal{F}_{\boldsymbol{\psi}} = \{\mathcal{F}_{\boldsymbol{\psi}} | \boldsymbol{\psi} \in \boldsymbol{\Psi}\}$ that hold the same parametrization form with $\mathcal{F}_{\boldsymbol{\psi}}$, where $\boldsymbol{\Psi} \subseteq \mathbf{R}^k$ is a parameter set that contains all possible parameter $\boldsymbol{\psi}$.

Behavior Formulation The behaviors in RL can be formulated as $\mu_{\boldsymbol{\theta},\mathbf{h},\boldsymbol{\psi}} = \mathcal{F}_{\boldsymbol{\psi}}(\boldsymbol{\Phi}_{\boldsymbol{\theta}_{1:\mathrm{M}},\mathbf{h}_{1:\mathrm{M}}})$, wherein $\boldsymbol{\Phi}_{\boldsymbol{\theta}_{1:\mathrm{M}},\mathbf{h}_{1:\mathrm{M}}}$ is a learned model vector containing M learned models, and $\mathcal{F}_{\boldsymbol{\psi}}$ is a M-input sampling strategy operator indexed by $\boldsymbol{\psi}$, wherein N is the number of learned models. Based on behavior formulation, we can formulate behavior space construction and behavior selection.

Definition 3.1 (Behavior Space Construction)**.** *Considering the RL problem that behaviors μ are generated from the learned model(s) $\boldsymbol{\Phi}_{\boldsymbol{\theta}_{1:\mathrm{M}},\mathbf{h}_{1:\mathrm{M}}}$ via a M-input sampling strategy operator $\mathcal{F}_{\boldsymbol{\psi}}$, all realizable behaviors can form a behavior space, which can be formulated as*

$$\mu_{\boldsymbol{\Theta},\mathbf{H},\boldsymbol{\Psi}} = \{\mu_{\boldsymbol{\theta},\mathbf{h},\boldsymbol{\psi}} = \mathcal{F}_{\boldsymbol{\psi}}(\boldsymbol{\Phi}_{\boldsymbol{\theta}_{1:\mathrm{M}},\mathbf{h}_{1:\mathrm{M}}}) | \boldsymbol{\theta} \in \boldsymbol{\Theta}, \mathbf{h} \in \mathbf{H}, \boldsymbol{\psi} \in \boldsymbol{\Psi}\}, \tag{2}$$

and when $\mathcal{F}_{\boldsymbol{\psi}}$ is a 1-input operator, it can be simplified into

$$\mu_{\boldsymbol{\Theta},\mathbf{H},\boldsymbol{\Psi}} = \{\mu_{\boldsymbol{\theta},\mathbf{h},\boldsymbol{\psi}} = \mathcal{F}_{\boldsymbol{\psi}}(\Phi_{\boldsymbol{\theta},\mathbf{h}}) | \boldsymbol{\theta} \in \boldsymbol{\Theta}, \mathbf{h} \in \mathbf{H}, \boldsymbol{\psi} \in \boldsymbol{\Psi}\}. \tag{3}$$

Definition 3.2 (Behavior Selection)**.** *The behavior selection problem can be formulated as finding a selection distribution $\mathcal{P}_{\mu_{\boldsymbol{\Theta},\mathbf{H},\boldsymbol{\Psi}}}$ to select the behaviors μ from behavior space $\mu_{\boldsymbol{\Theta},\mathbf{H},\boldsymbol{\Psi}}$ and maximizing some optimization target $\mathcal{L}_{\mathcal{P}}$, wherein $\mathcal{L}_{\mathcal{P}}$ is the optimization target of behavior selection.*

3.2 BEHAVIOR SPACE CONSTRUCTION

Unless otherwise specified, in this paper, we assume Assumption 1 holds. Under Assumption 1, equation 2 can be simplified into equation 4. According to equation 4, four core factors need to be considered when constructing a behavior space: the network structure Φ, the form of sampling strategy $\mathcal{F}$, the hyper-parameter set $\mathbf{H}$ and the parameter set $\mathbf{\Psi}$.

$$\mu_{\mathbf{H},\mathbf{\Psi}} = \{\mu_{\mathbf{h},\boldsymbol{\psi}} = \mathcal{F}_{\boldsymbol{\psi}}(\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{M}}})|\mathbf{h} \in \mathbf{H}, \boldsymbol{\psi} \in \mathbf{\Psi}\}. \tag{4}$$

Many notable representation learning approaches have explored how to design the network structure (Chen et al., 2021; Irie et al., 2021), but it is not the focus of our work. In this paper, we don't make any assumptions about the model structure, and we will discuss the behavior space on any given model structure, which means it can be applied to *any* model structure. The form of sampling strategy $\mathcal{F}$ can be selected according to the control task and algorithm. For instance, ϵ-greedy can be adopted in the value-based methods for discrete control task (Badia et al., 2020a); Boltzmann operator can be used in the policy-based methods for discrete control task (Espeholt et al., 2018); Gaussian distribution can be used in the continuous control task (Haarnoja et al., 2018). We also do not assume the specific parameterized form of the operator, and we only need to select an appropriate one according to the tasks at hand and selected algorithms. Given the model structure Φ and parametrization form of $\mathcal{F}$, the behavior space can be controlled by $\mathbf{\Psi}$ and $\mathbf{H}$. Diverse behavior spaces can be obtained by choosing different $\mathbf{\Psi}$ and $\mathbf{H}$. In the following, we will discuss more the behavior space design with respect to the property of $\mathcal{F}$ and the learned model vector $\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{M}}}$ in equation 4.

Single-Input Operator When $\mathcal{F}$ is a single-input operator, *e.g.*, ϵ-greedy, only one model can be used to generate the behavior. When there is only one learned model $\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{M}}} = \mathbf{\Phi}_{\mathbf{h}_{1:1}} = \Phi_{\mathbf{h}_1}$. When there is more than one learned model, since $\mathcal{F}$ can only take a single model each time., prior population-based methods adopt *model selection* to select a specific model $\Phi_{\mathbf{h}_j}$ from $\{\Phi_{\boldsymbol{\theta}_1,\mathbf{h}_1}, ..., \Phi_{\boldsymbol{\theta}_\mathrm{N},\mathbf{h}_\mathrm{N}}\}$ to generate a new behavior. Hence, $\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{M}}} = \mathbf{\Phi}_{\mathbf{h}_{1:1}} = \Phi_{\mathbf{h}_j}$ and $\mathbf{h}_j \in \mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_\mathrm{N}\}$. The drawback of this approach is that each behavior takes into account only one single model used to generate it but not the other models in the entire population. In this paper, we argue that a multi-input operator can tackle this problem.

Multi-Input Operator When $\mathcal{F}$ is a multi-input operator *e.g.*, a mixture model (Lindsay, 1995) (examples see equation 11), multiple learned models can be used to generate a behavior. When there is only one learned model or all the learned models overlapped, the input learned models is a M-D vector $\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{M}}} = (\Phi_{\mathbf{h}}, ..., \Phi_{\mathbf{h}})$, *i.e.*, $\mu = \mathcal{F}(\Phi_{\mathbf{h}}, ..., \Phi_{\mathbf{h}})$. In multiple-model scenario, when $\mathcal{F}$ is a M-input operator, the input learned models is a M-D vector $\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{M}}} = (\Phi_{\mathbf{h}_1}, ..., \Phi_{\mathbf{h}_\mathrm{M}})$ and $\mathbf{h}_j \in \mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_\mathrm{N}\}$. Furthermore, when $\mathrm{M} = \mathrm{N}$, all the learned models can be concluded while generating each behavior, and RL agents can automatically balance the exploration-exploitation trade-off of all learned models simply by selecting proper sampling strategy operator. However, studies on multi-input operators remain lacking. In this paper, we will propose a novel multi-input operator in Sec. 4.1.

3.3 BEHAVIOR SELECTION

According to equation 4, each behavior is uniquely indexed by $(\mathbf{h}, \boldsymbol{\psi})$, so we can cast the behavior selection into the selection of $(\mathbf{h}, \boldsymbol{\psi})$, which can be further simplified into the selection of $\mathbf{h}$ as Proposition 1 or $\boldsymbol{\psi}$ as Proposition 2. From these perspective, *the behavior selection is to select proper sampling strategy operators $\mathcal{F}_{\boldsymbol{\psi}}$ for learned model(s) $\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{M}}}$ to generate data-collection behaviors and maximize the target $\mathcal{L}_{\mathcal{P}}$.* Hence, there are two keys in behavior selection: **1)** Optimization Target $\mathcal{L}_{\mathcal{P}}$. **2)** The optimization algorithm to learn the selection distribution $\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}$ and maximize $\mathcal{L}_{\mathcal{P}}$. In this section, we will discuss them sequentially.

Optimization Target Two core factors have to be considered for the optimization target: the diversity-based measurement V_{μ}^{TD} (Eysenbach et al., 2019) and the value-based measurement

V_μ^{TV} (Parker-Holder et al., 2020). Integrating both, the optimization target can be formulated as

$$\begin{aligned}\mathcal{L}_{\mathcal{P}} &= \mathcal{R}_{\mu\sim\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}} + c\cdot\mathcal{D}_{\mu\sim\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}} \\ &= \mathbb{E}_{\mu\sim\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}}[V_\mu^{\mathrm{TV}} + c\cdot V_\mu^{\mathrm{TD}}],\end{aligned} \tag{5}$$

wherein, $\mathcal{R}_{\mu\sim\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}}$ and $\mathcal{D}_{\mu\sim\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}}$ is the expectation of value and diversity of behavior μ over the selection distribution $\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}$. When $\mathcal{F}_{\boldsymbol{\psi}}$ is unlearned and deterministic, behavior selection for each actor can be simplified into the selection of the learned models:

Proposition 1 (Learned Model Selection)**.** *When $\mathcal{F}_{\boldsymbol{\psi}}$ is a deterministic and one-to-one mapping for each actor at each training step (wall-clock), e.g., Agent57, equation 5 can be simplified into*

$$\mathcal{L}_{\mathcal{P}} = \mathbb{E}_{\mathbf{h}\sim\mathcal{P}_{\mathbf{H}}}\left[V_{\mu_{\mathbf{h}}}^{\mathrm{TV}} + c\cdot V_{\mu_{\mathbf{h}}}^{\mathrm{TD}}\right], \tag{6}$$

where $\mathcal{P}_{\mathbf{H}}$ is a selection distribution of $\mathbf{h}\in\mathbf{H}=\{\mathbf{h}_1,...,\mathbf{h}_{\mathrm{N}}\}$. For each actor, the behavior is generated from a selected learned model Φ_{h_i} and a deterministic $\mathcal{F}_{\boldsymbol{\psi}_j}$.

However, behavior selection via learned model selection may encounter the behavior circling problem:

Corollary 1 (Behavior Circling in Learned Model Selection)**.** *When the learned models overlapped as $\Phi_{\mathbf{h}_1}\approx ... \approx \Phi_{\mathbf{h}_{\mathrm{N}}}$ (See App. K), all realizable behavior of actor j will overlap as $\mathcal{F}_{\boldsymbol{\psi}_j}(\Phi_{\mathbf{h}_1})\approx ... \approx \mathcal{F}_{\boldsymbol{\psi}_j}(\Phi_{\mathbf{h}_{\mathrm{N}}})$. Behaviors from different model can not be distinguished by $\mathbf{h}_i$, the behavior selection via learned model selection becomes invalid and behavior control becomes fully rule-based.*

To handle the behavior circling, prior methods may adopt diversity-based multi-objective model training to boost the diversity of learned models. To avoid the harmful defects of multi-objective model training, in this paper, we attempt to achieve behavior selection via operator optimization:

Proposition 2 (Operator Optimization)**.** *When $\boldsymbol{\theta},\mathbf{h}$ are shared among behaviors[2], e.g., $\mu = \mathcal{F}_{\boldsymbol{\psi}}(\Phi_{\boldsymbol{\theta},\mathbf{h}})$ or $\mu=\mathcal{F}_{\boldsymbol{\psi}}(\Phi_{\boldsymbol{\theta}_1,\mathbf{h}_1},...,\Phi_{\boldsymbol{\theta}_{\mathrm{N}},\mathbf{h}_{\mathrm{N}}})$, equation 5 can be simplified into*

$$\mathcal{L}_{\mathcal{P}} = \mathbb{E}_{\boldsymbol{\psi}\sim\mathcal{P}_{\mathbf{\Psi}}}\left[V_{\mu_{\boldsymbol{\psi}}}^{\mathrm{TV}} + c\cdot V_{\mu_{\boldsymbol{\psi}}}^{\mathrm{TD}}\right], \tag{7}$$

where $\mathcal{P}_{\mathbf{\Psi}}$ is a selection distribution of $\boldsymbol{\psi}\in\mathbf{\Psi}$.

The operator optimization may be a cure for the behavior circling:

Corollary 2 (Operator Optimization Is An Antidote for Behavior Circling)**.** *As for an operator optimization method, the behavior of actor j is indexed by $\mathcal{F}_{\boldsymbol{\psi}}$. When all the learned models overlapped as $\Phi_{h_1}\approx ... \approx \Phi_{h_{\mathrm{N}}}$, the realizable behavior of actor j are*

$$\mathcal{F}_{\boldsymbol{\psi}_1}(\Phi_{h_1},...,\Phi_{h_{\mathrm{N}}}), \mathcal{F}_{\boldsymbol{\psi}_2}(\Phi_{h_1},...,\Phi_{h_{\mathrm{N}}}), ..., \mathcal{F}_{\boldsymbol{\psi}_\infty}(\Phi_{h_1},...,\Phi_{h_{\mathrm{N}}}), \tag{8}$$

wherein $\boldsymbol{\psi}_i$ is a continuous parameter. Assuming $\mathcal{F}_\psi$ is an injection, there are still infinite realizable behaviors for actor j.

Selection Distribution Optimization Given the optimization target $\mathcal{L}_{\mathcal{P}}$, we seek to find the optimal behavior selection distribution $\mathcal{P}_\mu^*$ that maximizes $\mathcal{L}_{\mathcal{P}}$:

$$\begin{aligned}\mathcal{P}^*_{\mu_{\mathbf{H},\mathbf{\Psi}}} := \mathrm{argmax}_{\mathcal{P}_{\mu_{\mathbf{H},\mathbf{\Psi}}}}\mathcal{L}_{\mathcal{P}} &\overset{(1)}{=} \mathrm{argmax}_{\mathcal{P}_{\mathbf{H}}}\mathcal{L}_{\mathcal{P}} \\ &\overset{(2)}{=} \mathrm{argmax}_{\mathcal{P}_{\mathbf{\Psi}}}\mathcal{L}_{\mathcal{P}},\end{aligned} \tag{9}$$

where (1) and (2) holds because we have Proposition 1 and 2, respectively. This optimization problem can be solved with existing optimizers, *e.g.*, evolutionary algorithm (Jaderberg et al., 2017), multi-arm bandits (MAB) (Badia et al., 2020a), etc.

3.4 Learnable Behavior Control

Integrating the behavior space construction and behavior selection into well-known RL methods, we can obtain a general framework for RL called **L**earnable **B**ehavioral **C**ontrol (**LBC**), which is concluded in App. E. Given any existing RL methods, by choosing different behavior spaces and behavior selection algorithms, we can obtain diverse novel methods based on LBC.

[2]Each behavior is generated from all the learned models and uniquely indexed by ψ.

3.5 AN LBC-BASED VERSION OF RL

The behavior space is vital for RL methods, which can be used to categorize RL algorithms. Given the model structure Φ, and the form of $\mathcal{F}$, the behavior space can be fully determined by $\mathbf{H}$ and $\mathbf{\Psi}$, which can be used to categorize RL methods. We say one algorithm belongs to LBC-$\mathbf{H}_{\mathrm{N}}^{\mathrm{M}}$-$\mathbf{\Psi}_{\mathrm{L}}^{\mathrm{K}}$ when **1)** the hyper-parameters $\mathbf{h}$ is a M-D vector and $\mathbf{h}$ has N possible values corresponding to N learned models, and **2)** $\boldsymbol{\psi}$ is a K-D vector and $\boldsymbol{\psi}$ has L possible values corresponding to L realizable sampling strategy operators at each training step. Based on that, we can offer a novel view to understand prior methods from the perspective of behavior control, which is illustrated in Tab. 2 of the App. F.

Example 1 (LBC-Based Version of Agent57)**.** *In Agent57, there are two hyper-parameters to index each learned model, so* $\mathbf{h} = (\gamma, \beta)$ *and* $\mathrm{M} = 2$*. There are 32 learned models corresponding to 32 possible values of* $\mathbf{h}$*, so* $\mathrm{N} = 32$*. Moreover, for each behavior,* $\mathcal{F}$ *is parameterized as the* ϵ*-greedy strategy, so* $\boldsymbol{\psi} = (\epsilon)$ *and* $\mathrm{K} = 1$*. Each* ϵ *takes values in a pre-defined set* $\{\epsilon_1, ..., \epsilon_{256}\}$ *at each training step, so* $\mathrm{L} = 256$*. Therefore, Agent57 belongs to* LBC-$\mathbf{H}_{32}^{2}$-$\mathbf{\Psi}_{256}^{I}$*. Moreover, Agent57 adopts learned model selection to achieve behavior selection from the behavior space.*

4 LBC-B: A BOLTZMANN BASED REALIZATION FOR LBC

In this section, we will use LBC to achieve behavior control of distributed off-policy actor-critic methods for Atari (Espeholt et al., 2018) via operator optimization in both single-input and multi-input scenarios. We uniformly adopt the model structure Φ as IMPALA deep (Espeholt et al., 2018) in both scenarios. As for the single-input scenario, we take the classic Boltzmann operator to parameterize $\mathcal{F}$. As for the multi-input scenario, we propose a multi-input Boltzmann operator to parameterize $\mathcal{F}$. Given Φ and $\mathcal{F}$, we can obtain diverse behavior spaces via choosing different $\mathbf{H}$ or $\mathbf{\Psi}$. With the behavior selection to choose behaviors from these behavior spaces, we can achieve behavior control at different granularities. Incorporating them into the distributed off-policy actor-critic methods, we can obtain a family of LBC-based algorithms, called LBC-$\mathcal{B}$-$\mathbf{H}_{\mathrm{N}}^{\mathrm{M}}$-$\mathbf{\Psi}_{\mathrm{L}}^{\mathrm{K}}$.

4.1 BOLTZMANN BASED BEHAVIOR SPACE CONSTRUCTION

Single-Input Operator In single-input scenario, to achieve behavior control for policy-based methods in discrete control task, we directly adopt the Boltzmann operator (Asadi & Littman, 2017) to parameterize the sampling strategy operator as $\mathcal{F}_{\boldsymbol{\psi}=\eta}(\Phi) = \mathrm{softmax}_{\eta}(\Phi) = \frac{e^{\eta\Phi}}{\sum_a e^{\eta\Phi}}$. Traversing $\eta \in (0, \infty)$, we can obtain infinite different $\mathcal{F}_{\eta}$. The behavior space can be formulated as

$$\mu_{\mathbf{H},\mathbf{\Psi}} = \{\mu_{\mathbf{h},\boldsymbol{\psi}} = \mathrm{softmax}_{\eta}(\Phi_{\mathbf{h}})\}, \tag{10}$$

wherein , $\mathbf{H} = \{\mathbf{h}\}$, $\mathbf{\Psi} = \{\boldsymbol{\psi} = \eta | \eta \in (0, \infty)\}$, Φ is a learned model indexed by $\mathbf{h}$, *e.g.*, $Q_{\mathbf{h}}$ (Song et al., 2019) or $A_{\mathbf{h}}$ (Wang et al., 2016). In this paper, we adopt $A_{\mathbf{h}}$. Adaptively fine-tuning η, we can adaptively control the trade-off of exploration and exploitation of a single learned model , *e.g.*, $\eta = 0$ for *no exploited* and $\eta = \infty$ for *fully exploited*.

Multi-Input Operator In multi-input scenario, we propose a multi-input Boltzmann operator as $\mathcal{F}_{\boldsymbol{\psi}}(\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{N}}}) = \sum_{i=1}^{\mathrm{N}} \zeta_i \, \mathrm{softmax}_{\eta_i}(\Phi_{\mathbf{h}_i})$ where $\mathbf{\Phi}_{\mathbf{h}_{1:\mathrm{N}}} = (\Phi_{\mathbf{h}_1}, ..., \Phi_{\mathbf{h}_{\mathrm{N}}})$, which can integrate all learned models together and *automatically choose how much to exploit or explore with respect to each learned model*. The behavior space can be formulated as

$$\mu_{\mathbf{H},\mathbf{\Psi}} = \{\mu_{\mathbf{h},\boldsymbol{\psi}} = \sum_{i=1}^{\mathrm{N}} \zeta_i \, \mathrm{softmax}_{\eta_i}(\Phi_{\mathbf{h}_i})\}, \tag{11}$$

wherein $\mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_{\mathrm{N}}\}$, and $\mathbf{\Psi} = \{\boldsymbol{\psi} = (\eta_1, \zeta_1, ..., \eta_{\mathrm{N}}, \zeta_{\mathrm{N}}) | \eta_i \in (0, \infty), \sum_{j=1}^{\mathrm{N}} \zeta_j = 1\}$, and N is the number of learned models.

4.2 MAB BASED BEHAVIOR SELECTION

According to Proposition 2, the behavior selection over behavior space 10 and 11 can be simplified to the selection of $\boldsymbol{\psi}$. That is, the behavior selection in both scenarios can be achieved in a unified manner, *i.e.*, finding the optimal $\boldsymbol{\psi}^* \in \mathbf{\Psi}$ that maximizes $\mathcal{L}_{\mathcal{P}}$ for each actor at each training step.

In this paper, we use MAB-based meta-controller to select $\boldsymbol{\psi} \in \boldsymbol{\Psi}$. As $\boldsymbol{\Psi}$ is a continuous multidimensional space, we discretize $\boldsymbol{\Psi}$ into K regions, and each region corresponds to an arm of MAB. At the beginning of a trajectory i, l-th actor will use MAB to sample an arm $k \in \mathcal{K} = \{1, 2, ..., \mathrm{K}\}$, corresponding to a parameter area indexed by k. Then, a specific index $\boldsymbol{\psi}$ will be uniformly sampled from the area, corresponding to a sampling strategy operator $\mathcal{F}_{\boldsymbol{\psi}}$. With $\mathcal{F}_{\boldsymbol{\psi}}$, we can obtain a behavior μ_k according to equation 10 or equation 11. Then, the l-th actor acts μ_k to obtain a trajectory τ_i and the undiscounted episode returns G_i, then G_i is used to update the reward model V_k of arm k, wherein the V_k is calculated by the expectation of the undiscounted episodic returns.

To handle the reward-diversity trade-off problem (Parker-Holder et al., 2020) in equation 7, we adopt Upper Confidence Bound (Garivier & Moulines, 2011) as

$$\mathrm{Score}_k = V_k + c \cdot \sqrt{\frac{\log(1 + \sum_{j \neq k}^{\mathrm{K}} N_j)}{1 + N_k}}, \tag{12}$$

wherein N_k means the number of visit of behavior μ_k. In equation 12, V_k is used to measure the value of each behavior, and the selected times N_k is used to avoid selecting the same arm repeatedly and ensure diverse behaviors could be selected.

To tackle the nonstationary problem (Badia et al., 2020a), we jointly train several bandits to form a population of bandits from very exploratory to purely exploitative and periodically update members of the population. For more details, see App. D.

5 Experiment

In this section, we designed our experiment to answer the following questions:

- Whether our methods can outperform prior SOTA RL algorithms in both sample efficiency and final performance (See Sec. 5.2 and Fig. 2)?
- Can our methods adaptively balance the exploration-exploration trade-off (See Fig. 3)?
- How to directly obtain novel algorithms via transforming the behavior space? What is the performance of methods with different behavior spaces (See Sec. 5.4)?
- How much performance will be degraded with a random behavior selection (See Fig. 4)?

5.1 Experimental Setup

5.1.1 Experimental Details

We showcase our performance in ALE (Bellemare et al., 2013). The standard pre-processing settings of Atari are identical to those of Agent57 (Badia et al., 2020a), and related parameters have been concluded in App. H. We employ a separate evaluation process to record scores continuously. We record the undiscounted episodic returns averaged over five seeds using a windowed mean over 32 episodes. To avoid any issues that aggregated metrics may have, App. I provides full learning curves for all games and detailed comparison tables of raw and normalized scores. For more experimental details, see App. G.

5.1.2 Implementation Details

We jointly train three models, and each model can be indexed by the hyper-parameters $\mathbf{h}_i = (\gamma_i, \mathfrak{R}_i)$, wherein $\mathfrak{R}_i$ is a reward shaping method (Badia et al., 2020a), and γ_i is a discounted factor. Each learned model $\Phi_{\mathbf{h}_i}$ adopts the dueling network structure (Wang et al., 2016), where $\Phi_{\mathbf{h}_i} = A_{\mathbf{h}_i} = Q_{\mathbf{h}_i} - V_{\mathbf{h}_i}$. More details of the network structure can be found in L. To correct for harmful discrepancy of off-policy learning, we adopt V-Trace (Espeholt et al., 2018) and ReTrace (Munos et al., 2016) to learn $V_{\mathbf{h}_i}$ and $Q_{\mathbf{h}_i}$. The policy is learned by policy gradient (Schulman et al., 2017). Based on equation 11, we could build a behavior space of three learned models as $\mu_{\mathbf{H},\boldsymbol{\Psi}} = \{\mu_{\mathbf{h},\boldsymbol{\psi}} = \sum_{i=1}^{3} \zeta_i \, \mathrm{softmax}_{\eta_i}(\Phi_{\mathbf{h}_i})\}$, wherein $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\}$, $\boldsymbol{\Psi} = \{\boldsymbol{\psi} = (\eta_1, \zeta_1, \eta_2, \zeta_2, \eta_3, \zeta_3) | \eta \in (0, \eta^+), \sum_{j=1}^{3} \zeta_j = 1\}$. The behavior selection is achieved by MAB described in Sec. 4.2, and more details can see App. D. Finally, we could obtain an off-policy actor-critic implementation of LBC called LBC-$\mathcal{B}$-H_3^2-$\boldsymbol{\Psi}_\infty^6$, which is our main algorithm.

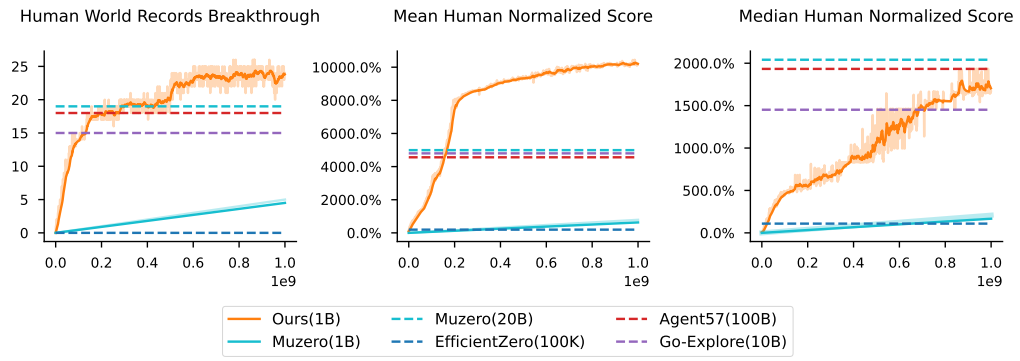

Figure 2: Learning Curve for ALE (Badia et al., 2020a).

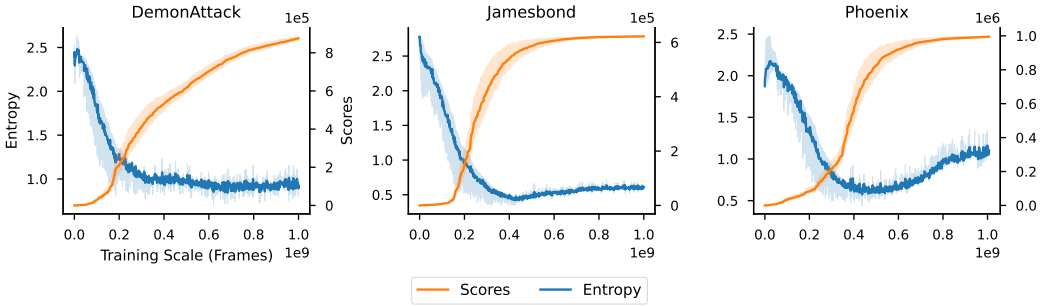

Figure 3: Behavior entropy and scores curve across training for different games where we achieved unprecedented performance. The names of the axes are the same as that of the leftmost figure.

5.2 Summary of Results

Results in ALE The aggregated results across games are reported in Fig. 2. From Fig. 2, EfficientZero (Ye et al., 2021) achieves remarkable learning efficiency in smaller data volumes but fails to obtain comparable final performance as other SOTA methods like Agent57 or Muzero. Among the algorithms with comparable final performance, our agents achieve the best mean HNS and surpass the most human world records across 57 games of the Atari benchmark with minimal training frames, leading to the best learning efficiency. Noting that Agent57 reported the maximum scores across training as the final score, and if we report our performance, in the same manner, our median is 1934%, which is **higher** than that of Agent57 and demonstrates our SOTA performance.

Discussion of Overall Results With LBC, we can understand why Agent57 could achieve high performance but relatively low learning efficiency. This is due to threefold reasons: **i)** Agent57 has a high-capacity behavior space ($N \times L$ possible behaviors) and a meta-controller to select which model to prioritize to generate behavior **ii)** to boost the exploration and behavior diversity, Agent57 introduced the intrinsic rewards into the training objective, but **iii)** multi-objective training typically requires more data to converge, or even not converge. Compared to Agent57, our success comes from threefold reasons: **i)** significantly increasing the number of realizable behaviors at each training step (approaching infinite), **ii)** a novel multi-input sampling strategy operator to automatically tackle the exploration and exploitation trade-off of all learned models simultaneously, **iii)** a meta-controller to balance the exploration-exploitation trade-off at each training step. Each of the proposed improvements will be succinctly studied in our ablation studies.

5.3 Case Study: Behavior Control

To further explore the mechanisms underlying the success of our methods, we adopt a case study to showcase our control process of behaviors. As shown in Fig. 3, in most tasks, our agents prefer exploratory behaviors first (i.e., high stochasticity policies with high entropy), and, as training progresses, the agents shift into producing experience from more exploitative behaviors. On the verge of peaking, the entropy of the behaviors could be maintained at a certain level (task-wise) instead of collapsing swiftly to zero to avoid converging prematurely to sub-optimal policies.

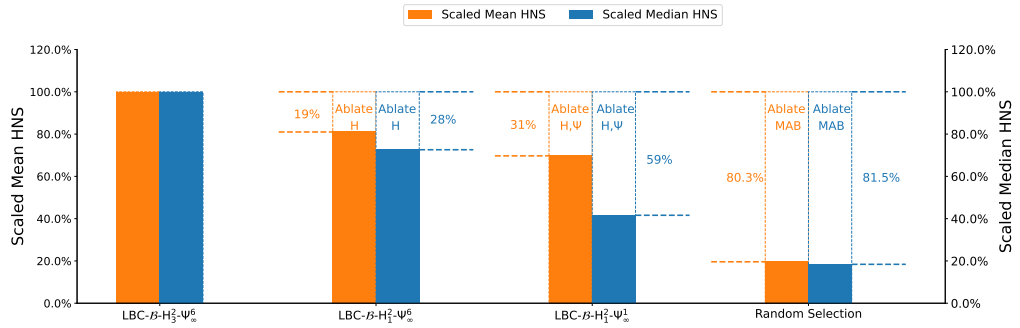

Figure 4: Ablation Results. All the results are scaled by that of LBC-$\mathcal{B}$-H_3^2-$\mathbf{\Psi}_\infty^6$ to improve readability.

5.4 Ablation Study

In this section, we investigate the effects of several properties of LBC. For more details, see App. J.

Behavior Space Decomposition To explore the effect of different behavior spaces, we decompose the behavior space of our main algorithms via reducing $\mathbf{H}$ and $\mathbf{\Psi}$:

1) Reducing $\mathbf{H}$. When we set all the learned models of LBC-$\mathcal{B}$-H_3^2-$\mathbf{\Psi}_\infty^6$ the same, the behavior space transforms from $\mathcal{F}(\Phi_{\mathbf{h}_1}, \Phi_{\mathbf{h}_2}, \Phi_{\mathbf{h}_3})$ into $\mathcal{F}(\Phi_{\mathbf{h}_1}, \Phi_{\mathbf{h}_1}, \Phi_{\mathbf{h}_1})$. $\mathbf{H}$ degenerates from $\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\}$ into $\{\mathbf{h}_1\}$. Keeping the other settings consistent with the main method, we can obtain an LBC-based method LBC-$\mathcal{B}$-H_1^2-$\mathbf{\Psi}_\infty^6$, whose behavior space is a sub-space of our main algorithm.

2) Reducing $\mathbf{\Psi}$. LBC-$\mathcal{B}$-H_1^2-$\mathbf{\Psi}_\infty^6$ still adopts a multi-input sampling strategy operator though every model input is the same. Based on that, we can directly adopt a single-input operator and build the behavior space as equation 10, where $\mathbf{\Psi}$ degenerates from $\{\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \boldsymbol{\psi}_3, \eta_1, \eta_2, \eta_3\}$ to $\{\eta\}$. Keeping the other settings consistent with the main method, we can obtain LBC-$\mathcal{B}$-H_1^2-$\mathbf{\Psi}_\infty^1$, whose behavior space is a sub-space of LBC-$\mathcal{B}$-H_1^2-$\mathbf{\Psi}_\infty^6$.

The performance of these methods is illustrated in Fig. 4, and from left to right, the behavior space of the first three algorithms decreases in turn (See Corollary 4 in App. C). It is evident that narrowing the behavior space via reducing $\mathbf{H}$ or $\mathbf{\Psi}$ will degrade the performance. On the contrary, the performance can be boosted by increasing the behavior space, which could be a promising way to improve the performance of existing methods.

Behavior Selection In this paper, we attempt to achieve behavior selection via operator optimization. To demonstrate that optimizing $\mathcal{F}$ can also boost the behavior diversity (See Corollary 2), we draw the t-SNE of visit states of our main algorithm and a method with rule-based $\mathcal{F}$ in App. J.3, where our main algorithm can acquire more diverse behaviors. Moreover, to highlight the importance of an appropriate behavior selection, we replace the meta-controller of our main algorithms with a random selection. The ablation results are illustrated in Fig. 4, from which it is evident that, with the same behavior space, not learning an appropriate selection distribution of behaviors will significantly degrade the performance.

6 Conclusion

In this paper, we try to handle the exploration-exploitation trade-off via behavior control. We present the first attempt to formulate behavior control from the learned models and the sampling strategy operator. We detail the behavior space construction of existing methods and how to improve them. We discuss behavior selection in two simplified cases: the learned model selection and operator optimization. Incorporating them into the existing RL methods, we obtain a scalable RL framework LBC. A branch of policy-based RL methods based on LBC has been proposed, which are then used to conduct a large number of experiments on Atari to demonstrate the effectiveness of LBC empirically. Apart from the full results, we do detailed ablation studies to examine the effectiveness of the proposed components. While there are many improvements and extensions to be explored going forward, we believe that the ability of LBC to enhance the optimization process of behaviors and to adapt to non-stationary learning problems results in a powerful platform to propel future research.

REFERENCES

Kavosh Asadi and Michael L. Littman. An alternative softmax operator for reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 243–252. PMLR, 2017. URL http://proceedings.mlr.press/v70/asadi17a.html.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 507–517. PMLR, 2020a. URL http://proceedings.mlr.press/v119/badia20a.html.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=Sye57xStvB.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613/jair.3912.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *CoRR*, abs/1901.10995, 2019. URL http://arxiv.org/abs/1901.10995.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. *SIAM J. Discret. Math.*, 17 (1):134–160, 2003. doi: 10.1137/S0895480102412856. URL https://doi.org/10.1137/S0895480102412856.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann (eds.), *Algorithmic Learning Theory - 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings*, volume 6925 of *Lecture Notes in Computer Science*, pp. 174–188. Springer, 2011. doi: 10.1007/978-3-642-24412-4_16. URL https://doi.org/10.1007/978-3-642-24412-4_16.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3215–3222. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Ronald A Howard. *Dynamic programming and markov processes.* John Wiley, 1960.

Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in Neural Information Processing Systems*, 34:7703–7717, 2021.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017. URL http://arxiv.org/abs/1711.09846.

Steven Kapturowski, Georg Ostrovski, John Quan, Rémi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

Bruce G Lindsay. Mixture models: theory, geometry, and applications. Ims, 1995.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *J. Artif. Intell. Res.*, 61:523–562, 2018. doi: 10.1613/jair.5699. URL https://doi.org/10.1613/jair.5699.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/mniha16.html.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1046–1054, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c3992e9a68c5ae12bd18488bc579b30d-Abstract.html.

Jack Parker-Holder, Aldo Pacchiano, Krzysztof Marcin Choromanski, and Stephen J. Roberts. Effective diversity in population based reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12,*

2020, virtual, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d1dc3a8270a6f9394f88847d7f0050cf-Abstract.html.

Carsten Lund Pedersen. Re: Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.05952.

Sepp Hochreiter; Jürgen Schmidhuber. Long short-term memory. *Neural Computation.*, 1997.

Simon Schmitt, Matteo Hessel, and Karen Simonyan. Off-policy actor-critic with shared experience replay. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8545–8554. PMLR, 2020. URL http://proceedings.mlr.press/v119/schmitt20a.html.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Zhao Song, Ronald Parr, and Lawrence Carin. Revisiting the softmax bellman operator: New benefits and new perspective. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5916–5925. PMLR, 2019. URL http://proceedings.mlr.press/v97/song19c.html.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3:9–44, 1988. doi: 10.1007/BF00115009. URL https://doi.org/10.1007/BF00115009.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6. URL https://www.worldcat.org/oclc/37293240.

Marin Toromanoff, Émilie Wirbel, and Fabien Moutarde. Is deep reinforcement learning really superhuman on atari? *CoRR*, abs/1908.04683, 2019. URL http://arxiv.org/abs/1908.04683.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782): 350–354, 2019. doi: 10.1038/s41586-019-1724-z. URL https://doi.org/10.1038/s41586-019-1724-z.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1995–2003. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/wangf16.html.

Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Mach. Learn.*, 8:279–292, 1992. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Marco A Wiering. *Explorations in efficient reinforcement learning*. PhD thesis, University of Amsterdam, 1999.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 25476–25488, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d5eca8dc3820cad9fe56a3bafda65ca1-Abstract.html.