Generalized Data Distribution Iteration

Jiajun Fan [1] **Changnan Xiao** [2]

Abstract

To obtain higher sample efficiency and superior final performance simultaneously has been one of the major challenges for deep reinforcement learning (DRL). Previous work could handle one of these challenges but typically failed to address them concurrently. In this paper, we try to tackle these two challenges simultaneously. To achieve this, we firstly decouple these challenges into two classic RL problems: data richness and exploration-exploitation trade-off. Then, we cast these two problems into the training data distribution optimization problem, namely to obtain desired training data within limited interactions, and address them concurrently via **i)** explicit modeling and control of the capacity and diversity of behavior policy and **ii)** more fine-grained and adaptive control of selective/sampling distribution of the behavior policy using a monotonic data distribution optimization. Finally, we integrate this process into Generalized Policy Iteration (GPI) and obtain a more general framework called **G**eneralized **D**ata Distribution **I**teration (GDI). We use the GDI framework to introduce operator-based versions of well-known RL methods from DQN to Agent57. Theoretical guarantee of the superiority of GDI compared with GPI is concluded. We also demonstrate our state-of-the-art (SOTA) performance on Arcade Learning Environment (ALE), wherein our algorithm has achieved 9620.33% mean human normalized score (HNS), 1146.39% median HNS and surpassed 22 human world records using only 200M training frames. Our performance is comparable to Agent57's while we consume 500 times less data. We argue that there is still a long way to go before obtaining real superhuman agents in ALE.

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University, Beijing, China [2]ByteDance, Beijing, China. Correspondence to: Changnan Xiao <xiaochangnan@bytedance.com>, Jiajun Fan <fanjj21@mails.tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

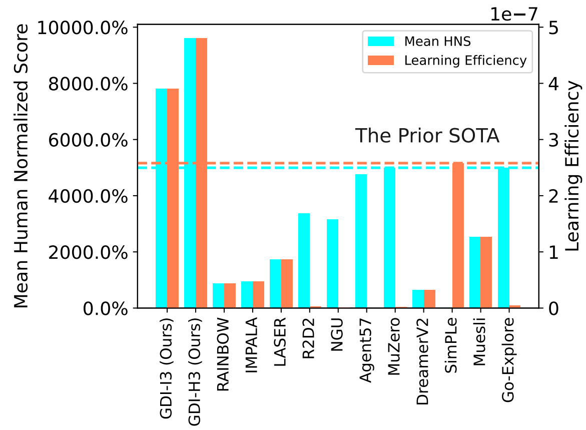
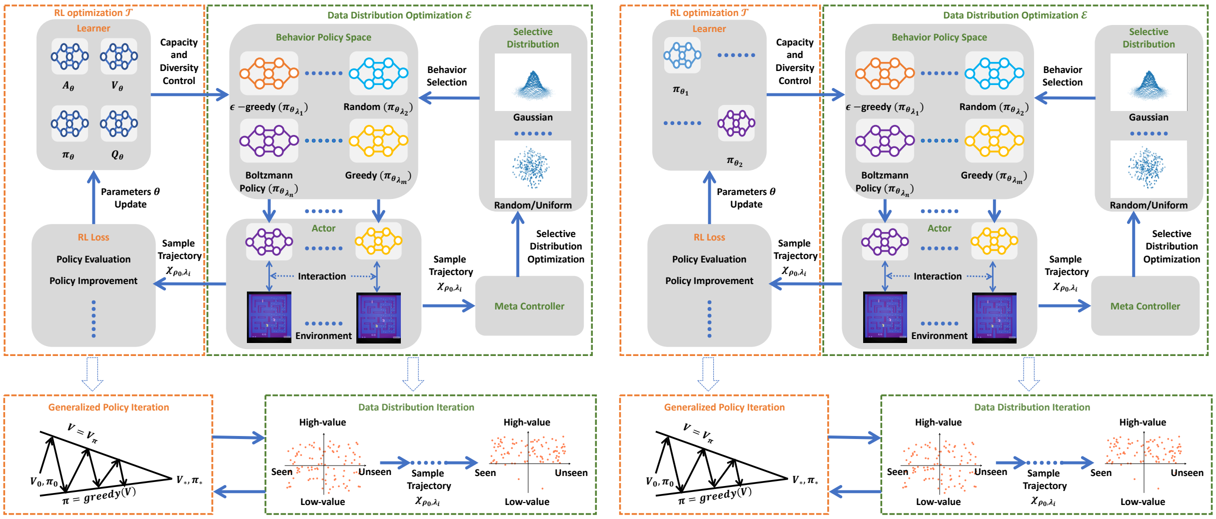


Figure 1. Performance of algorithms of Atari 57 games on mean HNS(%) and corresponding learning/sample efficiency calculated by $\frac{\text{Mean HNS}}{\text{Training Scale (frames)}}$. For more benchmark results, can see App. J.

1. Introduction

Reinforcement learning (RL) algorithms, when combined with high-capacity deep neural networks, have shown promise in domains ranging from video games (Mnih et al., 2015) to robotic manipulation (Schulman et al., 2015; 2017b). However, it still suffers from high sample complexity and unsatisfactory final performance, especially compared to human learning (Tsividis et al., 2017). Prior work could handle one of these problems but commonly failed to tackle both of them simultaneously.

Model-free RL methods typically obtain remarkable final performance via finding a way to encourage exploration and improve the data richness (e.g., $\frac{\text{Seen Conditions}}{\text{All Conditions}}$) that guarantees traversal of *all possible* conditions. These methods (Ecoffet et al., 2019; Badia et al., 2020a) could perform remarkably well when interactions are (nearly) *limitless* but normally fail when interactions are *limited*. We argue that when interactions are *limited*, finding a way to guarantee traversal of *all unseen* conditions is unreasonable, and perhaps we should find a way to traverse the *nontrivial* conditions (e.g., unseen (Ecoffet et al., 2019) and high-value (Kumar et al., 2020)) first and avoid traversing the *trivial/low-value* conditions repeatedly. In other words, we should explicitly control the training data distribution in RL and maximize the probabil-



(a) Isomorphism GDI

(b) Heterogeneous GDI

Figure 2. Algorithm Architecture Diagram. **(a)** The Isomorphism architecture of GDI, wherein the behavior policy space (e.g., the soft entropy policy space, $\pi_{\theta_\lambda} = \epsilon \cdot \text{Softmax}\left(\frac{A_{\theta_1}}{\tau_1}\right) + (1-\epsilon) \cdot \text{Softmax}\left(\frac{A_{\theta_2}}{\tau_2}\right)$) is constructed by the base policy with shared parameters (i.e., $\theta_1 = \theta_2 = \theta$) and indexed by $\lambda = (\tau_1, \tau_2, \epsilon)$. **(b)** The Heterogeneous architecture of GDI, wherein the behavior policy space is constructed by the base policy with different parameters (i.e., $\theta_1 \neq \theta_2$) and indexed by λ. For more details, can see Sec. 4 and 5.

ity of nontrivial conditions being traversed, namely the *data distribution optimization* (see Fig. 2).

In RL, training data distribution is normally controlled by the behavior policy (Sutton & Barto, 2018; Mnih et al., 2015), so that the data richness can be controlled by the capacity and diversity of the behavior policy. Wherein the capacity describes *how many different behavior policies there are in the policy space*, and the diversity describes *how many different behavior policies are selected/sampled from the policy space to generate training data* (discussed in Sec. 4.2). When interactions are limitless, increasing the capacity and maximizing the diversity via randomly sampling behavior policies (most prior works have achieved SOTA in this way) can significantly improve the data richness and guarantee traversal of almost all *unseen* conditions, which induces better final performance (Badia et al., 2020a) and generalization (Ghosh et al., 2021). However, perhaps surprisingly, this is not the case when interactions are limited, where each interaction is rare and the selection of the behavior policy becomes important. In conclusion, we should increase the probability of the traversal of *unseen* conditions (i.e., exploration) via increasing the *capacity* and *diversity* of the behavior policy and maximize the probability of *high-value* conditions (i.e., exploitation) being traversed via optimizing the selective distribution of the behavior policy. It's also known as the exploration-exploitation trade-off problem.

From this perspective, we can understand why the prior SOTA algorithms, such as Agent57 and Go-Explore, failed to obtain high sample efficiency. They have collected massive data to guarantee the traversal of *unseen* conditions but ignore the different values of data. Therefore, they wasted many trials to collect *useless/low-value* data, which accounts for their low sample efficiency. In other words, they failed to tackle the data distribution optimization problem.

In this paper, we argue that the sample efficiency of model-free methods can be significantly improved (even outperform the SOTA model-based schemes (Hafner et al., 2020)) without degrading the final performance via data distribution optimization. To achieve this, we propose a data distribution optimization operator $\mathcal{E}$ to iteratively optimize the selective distribution of the behavior policy and thereby optimize the training data distribution. Specifically, we construct a parameterized policy space indexed by λ called the soft entropy space, which enjoys a larger capacity than Agent57. The behavior policies are sampled from this policy space via a sampling distribution. Then, we adopt a meta-learning method to optimize the sampling distribution of behavior policies iteratively and thereby achieve a more fine-grained exploration and exploitation trade-off. Moreover, training data collected by the optimized behavior policies will be used for RL optimization via the operator $\mathcal{T}$. This process will be illustrated in Fig. 2, generalized in Sec. 4, proved superior in Sec. 4.6 and implemented in Sec. 5.1.

The main contributions of our work are:

1. **A General RL Framework.** *Efficient learning* within

limited interactions induces the data distribution optimization problem. To tackle this problem, we firstly explicitly control the diversity and capacity of the behavior policy (see Sec. 4.2) and then optimize the sampling distribution of behavior policies iteratively via a data distribution optimization operator (see Sec. 4.4). After integrating them into GPI, we obtain a general RL framework, GDI (see Fig. 2).

2. **An Operator View of RL Algorithms.** We use the GDI framework to introduce operator-based versions of well-known RL methods from DQN to Agent57 in Sec. 4.5, which leads to a better understanding of their original counterparts.

3. **Theoretical Proof of Superiority.** We offer theoretical proof of the superiority of GDI in the case of both first-order and second-order optimization in Sec. 4.6.

4. **The State-Of-The-Art Performance.** From Fig. 1, our algorithm GDI-H^3 has achieved 9620.33% mean HNS, outperforming the SOTA model-free algorithms Agent57. Surprisingly, our learning efficiency has outperformed the SOTA model-based methods Muzero and Dreamer-V2. Furthermore, our method has surpassed 22 Human World Records in 38 playtime days.

2. Related Work

Data richness. As claimed by (Ghosh et al., 2021), generalization to unseen test conditions from a limited number of training conditions induces implicit partial observability, effectively turning even fully observed MDPs into POMDPs, which makes generalization in RL much more difficult. Therefore, data richness (e.g., $\frac{\text{Seen Conditions}}{\text{All Conditions}}$) is vital for the generalization and performance of RL agents. When interactions are limited, more diverse behavior policies increase the data richness and thereby reduce the proportion of unseen conditions and improve generalization and performance. Therefore, we can recast this problem into the problem to control the capacity and diversity of the behavior policy. There are two promising ways to handle this issue. Firstly, some RL methods adopt intrinsic reward to encourage exploration, where unsupervised objectives, auxiliary tasks and other techniques induce the intrinsic reward (Pathak et al., 2017). Other methods (Badia et al., 2020a) introduced a diversity-based regularizer into the RL objective and trained a family of policies with different degrees of exploratory behaviors. Despite both obtaining SOTA performance, adopting intrinsic rewards and entropy regularization has increased the uncertainty of environmental transition. We argue that the inability to effectively tackle the data distribution optimization accounts for their low learning efficiency.

Exploration and exploitation trade-off. Exploration and exploitation trade-off remains one of the significant challenges in DRL (Badia et al., 2020b; Sutton & Barto, 2018). In general, methods that guarantee to find an optimal policy require the number of visits to each state–action pair to approach infinity. The entropy of policy would collapse to zero swiftly after a finite number of steps may never learn to act optimally; they may instead converge prematurely to suboptimal policies and never gather the data they need to learn to act optimally. Therefore, to ensure that all state-action pairs are encountered infinitely, off-policy learning methods are widely used (Mnih et al., 2016; Espeholt et al., 2018), and agents must learn to adjust the entropy (exploitation degree) of the behavior policy. Adopting stochastic policies into the behavior policy has been widely used in RL algorithms (Mnih et al., 2015; Hessel et al., 2017), such as the ϵ-greedy (Watkins, 1989). These methods can perform remarkably well in dense reward scenarios (Mnih et al., 2015), but fail to learn in sparse reward environments. Recent approaches (Badia et al., 2020a) have proposed to train a family of policies and provide intrinsic rewards and entropy regularization to agents to drive exploration. Among these methods, the intrinsic rewards are proportional to some notion of saliency, quantifying how different the current state is from those already visited. They have achieved SOTA performance at the cost of a relatively lower sample efficiency. We argue that these algorithms overemphasize the role of exploration to traverse *unseen* conditions but ignore the *value* of data and thereby waste many trails to collect *low-value* data, accounting for their low sample efficiency.

3. Preliminaries

The RL problem 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 $\rho_0(s) : \mathcal{S} \to \Delta(\mathcal{S})$, where we use Δ to represent the probability simplex. At each time t, the agent chooses an action $a_t \in \mathcal{A}$ according to the policy $\pi(a_t|s_t) : \mathcal{S} \to \Delta(\mathcal{A})$ 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 $p(s' \mid s, a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$. The process continues until the agent reaches a terminal state or a maximum time step. Define the discounted state visitation distribution as $d_{\rho_0}^{\pi}(s) = (1-\gamma)\mathbf{E}_{s_0 \sim \rho_0}\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{P}(s_t = s|s_0)\right]$. The goal of reinforcement learning is to find the optimal policy π^* that maximizes the expected sum of discounted rewards, denoted by $\mathcal{J}$ (Sutton & Barto, 2018):

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

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

4. Methodology

4.1. Notation Definition

Let's introduce our notations first, which are also summarized in App. A.

Define Λ to be an index set, $\Lambda \subseteq \mathbf{R}^k$. $\lambda \in \Lambda$ is an index in Λ. $(\Lambda, \mathcal{B}|_\Lambda, \mathcal{P}_\Lambda)$ is a probability space, where $\mathcal{B}|_\Lambda$ is a Borel σ-algebra restricted to Λ. Under the setting of meta-RL, Λ can be regarded as the set of all possible meta information. Under the setting of population-based training (PBT) (Jaderberg et al., 2017), Λ can be regarded as the set of the whole population.

Define Θ to be a set of all possible values of parameters (e.g., parameters of value function network and policy network). $\theta \in \Theta$ is some specific value of parameters. For each index λ, there exists a specific mapping between each parameter of θ and λ, denoted as θ_λ, to indicate the parameters in θ corresponding to λ (e.g., ϵ in ϵ-greedy behavior policies). Under the setting of linear regression $y = w \cdot x$, $\Theta = \{w \in R^n\}$ and $\theta = w$. If λ represents using only the first half features to perform regression, assume $w = (w_1, w_2)$, then $\theta_\lambda = w_1$. Under the setting of RL, θ_λ defines a parameterized policy indexed by λ, denoted as π_{θ_λ}.

Define $\mathcal{D} \overset{def}{=} \{d_{\rho_0}^{\pi} \,|\, \pi \in \Delta(\mathcal{A})^{\mathcal{S}}, \rho_0 \in \Delta(\mathcal{S})\}$ to be the set of all states visitation distributions. For the parameterized policies, denote $\mathcal{D}_{\Lambda,\Theta,\rho_0} \overset{def}{=} \{d_{\rho_0}^{\pi_{\theta_\lambda}} \,|\, \theta \in \Theta, \lambda \in \Lambda\}$. Note that $(\Lambda, \mathcal{B}|_\Lambda, \mathcal{P}_\Lambda)$ is a probability space on Λ, which induces a probability space on $\mathcal{D}_{\Theta,\Lambda,\rho_0}$, with the probability measure given by $\mathcal{P}_{\mathcal{D}}(\mathcal{D}_{\Lambda_0,\Theta,\rho_0}) = \mathcal{P}_\Lambda(\Lambda_0),\ \forall \Lambda_0 \in \mathcal{B}|_\Lambda$.

We use x to represent one sample, which contains all necessary information for learning. As for DQN, $x = (s_t, a_t, r_t, s_{t+1})$. As for R2D2, $x = (s_t, a_t, r_t, \ldots, s_{t+N}, a_{t+N}, r_{t+N}, s_{t+N+1})$. As for IMPALA, x also contains the distribution of the behavior policy. The content of x depends on the algorithm, but it's assumed to be sufficient for learning. We use $\mathcal{X}$ to represent the set of samples. At training stage t, given the parameter $\theta = \theta^{(t)}$, the distribution of the index set $\mathcal{P}_\Lambda = \mathcal{P}_\Lambda^{(t)}$ (e.g., sampling distribution of behavior policy) and the distribution of the initial state ρ_0, we denote the set of samples as

$$\begin{aligned}\mathcal{X}_{\rho_0}^{(t)} &\overset{def}{=} \bigcup_{d_{\rho_0}^{\pi} \sim \mathcal{P}_{\mathcal{D}}^{(t)}} \{x | x \sim d_{\rho_0}^{\pi}\} \\ &= \bigcup_{\lambda \sim \mathcal{P}_\Lambda^{(t)}} \{x | x \sim d_{\rho_0}^{\pi_\theta}, \theta = \theta_\lambda^{(t)}\} \\ &\triangleq \bigcup_{\lambda \sim \mathcal{P}_\Lambda^{(t)}} \mathcal{X}_{\rho_0,\lambda}^{(t)}.\end{aligned}$$

4.2. Capacity and Diversity Control of Behavior Policy

We consider the problem that behavior policies μ are sampled from a policy space $\{\pi_{\theta_\lambda} | \lambda \in \Lambda\}$ which is parameterized by the policy network and indexed by the index set Λ. The capacity of μ describes *how many different behavior policies are there in the policy space*, controlled by the base policy's capacity (e.g., shared parameters or not) and the size of the index set $|\Lambda|$. Noting that there are two sets of parameters, namely λ and θ. The diversity describes *how many different behavior policies are actually selected from the policy space to generate training data*, controlled by the sampling/selective distribution $\mathcal{P}_\Lambda$ (see Fig. 2).

After the capacity of the base policy is determined, we can explicitly control the data richness via the size of the index set and the sampling distribution $\mathcal{P}_\Lambda$. On the condition that interactions are limitless, increasing the size of the index set can significantly improve the data richness and thus is more important for a superior final performance since the diversity can be maximized via adopting a uniform distribution (most prior works have achieved SOTA in this way). However, it's data inefficient and the condition may never hold. Considering interactions are limited, the optimization of the sampling distribution, namely to select suitable behavior policies to generate training data, is crucial for sample efficiency because each interaction is rare. It's also known as the exploration-exploitation trade-off problem.

4.3. Data Distribution Optimization Problem

In conclusion, the final performance can be significantly improved via increasing the data richness controlled by the capacity and diversity of behavior policy. The sample efficiency is significantly influenced by the exploration-exploitation trade-off, namely the sampling/selective distribution of the behavior policy. In general, on the condition that the capacity of behavior policy is *determined* and training data is totally generated by behavior policies, these problems can be cast into the data distribution optimization problem:

Definition 4.1 (Data Distribution Optimization Problem)**.** *Finding a selective distribution $\mathcal{P}_\Lambda$ that samples behavior policies π_{θ_λ} from a parameterized policy space that indexed by Λ and maximizing some target function $L_{\mathcal{E}}$, where the $L_{\mathcal{E}}$ can be any target function (e.g., RL target) that describes what kind of data do agents desire (i.e., a measure of the importance/value of the sample trajectory).*

4.4. Generalized Data Distribution Iteration

Now we introduce our main algorithm to handle the data distribution optimization problem in RL.

$\mathcal{T}$ defined as $\theta^{(t+1)} = \mathcal{T}(\theta^{(t)}, \{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda \sim \mathcal{P}_\Lambda^{(t)}})$ is a typical optimization operator of RL algorithms, which utilizes the

Algorithm 1 Generalized Data Distribution Iteration

Initialize Λ, Θ, $\mathcal{P}_\Lambda^{(0)}$, $\theta^{(0)}$.
for $t = 0, 1, 2, \dots$ **do**
  Sample $\{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}$. {Data Sampling}
  $\theta^{(t+1)} = \mathcal{T}(\theta^{(t)}, \{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}})$. {Generalized Policy Iteration}
  $\mathcal{P}_\Lambda^{(t+1)} = \mathcal{E}(\mathcal{P}_\Lambda^{(t)}, \{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}})$. {Data Distribution Iteration}
end for

collected samples to update the parameters for maximizing some function $L_\mathcal{T}$. For instance, $L_\mathcal{T}$ may contain the policy gradient and the state value evaluation for the policy-based methods, may contain generalized policy iteration for the value-based methods, and may also contain some auxiliary tasks or intrinsic rewards for specially designed methods.

$\mathcal{E}$ defined as $\mathcal{P}_\Lambda^{(t+1)} = \mathcal{E}(\mathcal{P}_\Lambda^{(t)}, \{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}})$ is a data distribution optimization operator. It uses the samples $\{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}$ to update $\mathcal{P}_\Lambda$ and maximize some function $L_\mathcal{E}$, namely,

$$\mathcal{P}_\Lambda^{(t+1)} = \arg\max_{\mathcal{P}_\Lambda} L_\mathcal{E}(\{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda}).$$

Since $\mathcal{P}_\Lambda$ is parameterized, we abuse the notation and use $\mathcal{P}_\Lambda$ to represent the parameter of $\mathcal{P}_\Lambda$. If $\mathcal{E}$ is a first-order optimization operator, then we can write $\mathcal{E}$ explicitly as

$$\mathcal{P}_\Lambda^{(t+1)} = \mathcal{P}_\Lambda^{(t)} + \eta\nabla_{\mathcal{P}_\Lambda^{(t)}} L_\mathcal{E}(\{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}).$$

If $\mathcal{E}$ is a second-order optimization operator, like natural gradient, we can write $\mathcal{E}$ formally as

$$\mathcal{P}_\Lambda^{(t+1)} = \mathcal{P}_\Lambda^{(t)} + \eta\mathbf{F}(\mathcal{P}_\Lambda^{(t)})^\dagger\nabla_{\mathcal{P}_\Lambda^{(t)}} L_\mathcal{E}(\{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}),$$
$$\mathbf{F}(\mathcal{P}_\Lambda^{(t)}) = \left[\nabla_{\mathcal{P}_\Lambda^{(t)}} \log\mathcal{P}_\Lambda^{(t)}\right]\cdot\left[\nabla_{\mathcal{P}_\Lambda^{(t)}} \log\mathcal{P}_\Lambda^{(t)}\right]^\top,$$

where $\dagger$ denotes the Moore-Penrose pseudoinverse of the matrix.

4.5. An Operator View of RL Methods

We can further divide all algorithms into two categories, GDI-I^n and GDI-H^n. n represents the degree of freedom of Λ, which is the dimension of selective distribution. I represents Isomorphism. We say one algorithm belongs to GDI-I^n, if $\theta = \theta_\lambda, \forall\lambda\in\Lambda$. H represents Heterogeneous. We say one algorithm belongs to GDI-H^n, if $\theta_{\lambda_1} \neq \theta_{\lambda_2}, \exists\lambda_1,\lambda_2\in\Lambda$. By definition, GDI-H^n is a much larger set than GDI-I^n, but many algorithms belong to GDI-I^n rather than GDI-H^n. We say one algorithm is "w/o $\mathcal{E}$" if it doesn't contain the operator $\mathcal{E}$, which means its $\mathcal{E}$ is an identical mapping and the data distribution is not additionally optimized. Now, we could understand some well-known RL methods from the view of GDI.

For DQN, RAINBOW, PPO and IMPALA, they are in GDI-I^0 w/o $\mathcal{E}$. Let $|\Lambda| = 1$, WLOG, assume $\Lambda = \{\lambda_0\}$. Then, the probability measure $\mathcal{P}_\Lambda$ collapses to $\mathcal{P}_\Lambda(\lambda_0) = 1$. $\Theta = \{\theta_{\lambda_0}\}$. $\mathcal{E}$ is an identical mapping of $\mathcal{P}_\Lambda^{(t)}$. $\mathcal{T}$ is the first-order operator that optimizes the loss functions.

For Ape-X and R2D2, they are in GDI-I^1 w/o $\mathcal{E}$. Let $\Lambda = \{\epsilon_l | \, l = 1,\dots,256\}$. $\mathcal{P}_\Lambda$ is uniform, $\mathcal{P}_\Lambda(\epsilon_l) = |\Lambda|^{-1}$. Since all actors and the learner share parameters, we have $\theta_{\epsilon_1} = \theta_{\epsilon_2}$ for $\forall\epsilon_1,\epsilon_2\in\Lambda$, hence $\Theta = \bigcup_{\epsilon\in\Lambda}\{\theta_\epsilon\} = \{\theta_{\epsilon_l}\}$, $\forall\, l = 1,\dots,256$. $\mathcal{E}$ is an identical mapping, because $\mathcal{P}_\Lambda^{(t)}$ is always a uniform distribution. $\mathcal{T}$ is the first-order operator that optimizes the loss functions.

For LASER, it's in GDI-H^1 w/o $\mathcal{E}$. Let $\Lambda = \{i | \, i = 1,\dots,K\}$ to be the number of learners. $\mathcal{P}_\Lambda$ is uniform, $\mathcal{P}_\Lambda(i) = |\Lambda|^{-1}$. Since different learners don't share parameters, $\theta_{i_1}\cap\theta_{i_2} = \emptyset$ for $\forall i_1, i_2\in\Lambda$, hence $\Theta = \bigcup_{i\in\Lambda}\{\theta_i\}$. $\mathcal{E}$ is an identical mapping. $\mathcal{T}$ can be formulated as a union of $\theta_i^{(t+1)} = \mathcal{T}_i(\theta_i^{(t)}, \{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}})$, which represents optimizing θ_i of the ith learner with shared samples from other learners.

For PBT, it's in GDI-H^{n+1}, where n is the number of searched hyperparameters. Let $\Lambda = \{h\}\times\{i|i = 1,\dots,K\}$, where h represents the hyperparameters being searched and K is the population size. $\Theta = \bigcup_{i=1,\dots,K}\{\theta_{i,h}\}$, where $\theta_{i,h_1} = \theta_{i,h_2}$ for $\forall(h_1,i),(h_2,i)\in\Lambda$. $\mathcal{E}$ is the meta-controller that adjusts h for each i, which can be formally written as $\mathcal{P}_\Lambda^{(t+1)}(\cdot,i) = \mathcal{E}_i(\mathcal{P}_\Lambda^{(t)}(\cdot,i), \{\mathcal{X}_{\rho_0,(h,i)}^{(t)}\}_{h\sim\mathcal{P}_\Lambda^{(t)}(\cdot,i)})$, which optimizes $\mathcal{P}_\Lambda$ according to the performance of all agents in the population. $\mathcal{T}$ can also be formulated as a union of $\mathcal{T}_i$, but is $\theta_i^{(t+1)} = \mathcal{T}_i(\theta_i^{(t)}, \{\mathcal{X}_{\rho_0,(h,i)}^{(t)}\}_{h\sim\mathcal{P}_\Lambda^{(t)}(\cdot,i)})$, which represents optimizing the ith agent with samples from the ith agent.

For NGU and Agent57, it's in GDI-I^2. Let $\Lambda = \{\beta_i|i = 1,\dots,m\}\times\{\gamma_j|j = 1,\dots,n\}$, where β is the weight of the intrinsic value function and γ is the discount factor. Since all actors and the learner share variables, $\Theta = \bigcup_{(\beta,\gamma)\in\Lambda}\{\theta_{(\beta,\gamma)}\} = \{\theta_{(\beta,\gamma)}\}$ for $\forall(\beta,\gamma)\in\Lambda$. $\mathcal{E}$ is an optimization operator of a multi-arm bandit controller with UCB, which aims to maximize the expected cumulative rewards by adjusting $\mathcal{P}_\Lambda$. Different from above, $\mathcal{T}$ is identical to our general definition $\theta^{(t+1)} = \mathcal{T}(\theta^{(t)}, \{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}})$, which utilizes samples from all λs to update the shared θ.

For Go-Explore, it's in GDI-H^1. Let $\Lambda = \{\tau\}$, where τ

represents the stopping time of switching between robustification and exploration. $\Theta = \{\theta_r\} \cup \{\theta_e\}$, where θ_r is the robustification model and θ_e is the exploration model. $\mathcal{E}$ is a search-based controller, which defines the next $\mathcal{P}_\Lambda$ for better exploration. $\mathcal{T}$ can be decomposed into $(\mathcal{T}_r, \mathcal{T}_e)$.

4.6. Monotonic Data Distribution Optimization

We see that many algorithms can be formulated as a special case of GDI. For algorithms without a meta-controller, whose data distribution optimization operator $\mathcal{E}$ is an identical mapping, the guarantee that the learned policy could converge to the optimal policy has been widely studied, for instance, GPI in (Sutton & Barto, 2018) and policy gradient in (Agarwal et al., 2019). However, for algorithms with a meta-controller, whose data distribution optimization operator $\mathcal{E}$ is non-identical, though most algorithms in this class show superior performance, it still lacks a general study on why the data distribution optimization operator $\mathcal{E}$ helps. In this section, with a few assumptions, we show that given the same optimization operator $\mathcal{T}$, a GDI with a non-identical data distribution optimization operator $\mathcal{E}$ is always superior to that without $\mathcal{E}$.

For brevity, we denote the expectation of $L_\mathcal{E}, L_\mathcal{T}$ for each $\lambda \in \Lambda$ as $\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda)$ and $\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)$, calculated as

$$\begin{aligned}\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda) &= \mathbf{E}_{x\sim\pi_{\theta_\lambda}}[L_\mathcal{E}(\{\mathcal{X}_{\rho_0,\lambda}\})]\\ \mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda) &= \mathbf{E}_{x\sim\pi_{\theta_\lambda}}[L_\mathcal{T}(\{\mathcal{X}_{\rho_0,\lambda}\})]\end{aligned}$$

and denote the expectation of $\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda), \mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)$ for any $\mathcal{P}_\Lambda$ as $\mathcal{L}_\mathcal{E}(\mathcal{P}_\Lambda, \theta) = \mathbf{E}_{\lambda\sim\mathcal{P}_\Lambda}[\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda)]$, $\mathcal{L}_\mathcal{T}(\mathcal{P}_\Lambda, \theta) = \mathbf{E}_{\lambda\sim\mathcal{P}_\Lambda}[\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)]$.

Assumption 1 (Uniform Continuous Assumption)**.** *For $\forall \epsilon > 0$, $\forall s \in \mathcal{S}$, $\exists\, \delta > 0$, $s.t. |V^{\pi_1}(s) - V^{\pi_2}(s)| < \epsilon$, $\forall\, d_\pi(\pi_1, \pi_2) < \delta$, where d_π is a metric on $\Delta(\mathcal{A})^\mathcal{S}$. If π is parameterized by θ, then for $\forall \epsilon > 0$, $\forall s \in \mathcal{S}$, $\exists\, \delta > 0$, $s.t. |V^{\pi_{\theta_1}}(s) - V^{\pi_{\theta_2}}(s)| < \epsilon$, $\forall ||\theta_1 - \theta_2|| < \delta$.*

Remark. *(Dadashi et al., 2019) shows V^π is infinitely differentiable everywhere on $\Delta(\mathcal{A})^\mathcal{S}$ if $|\mathcal{S}| < \infty, |\mathcal{A}| < \infty$. (Agarwal et al., 2019) shows V^π is β-smooth, namely bounded second-order derivative, for direct parameterization. If $\Delta(\mathcal{A})^\mathcal{S}$ is compact, continuity implies uniform continuity.*

Assumption 2 (Formulation of $\mathcal{E}$ Assumption)**.** *Assume $\mathcal{P}_\Lambda^{(t+1)} = \mathcal{E}(\mathcal{P}_\Lambda^{(t)}, \{\mathcal{X}_{\rho_0,\lambda}^{(t)}\}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}})$ can be written as $\mathcal{P}_\Lambda^{(t+1)}(\lambda) = \mathcal{P}_\Lambda^{(t)}(\lambda)\frac{\exp(\eta\mathcal{L}_\mathcal{E}(\lambda,\theta_\lambda^{(t)}))}{Z^{(t+1)}}$, $Z^{(t+1)} = \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}[\exp(\eta\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda^{(t)}))]$.*

Remark. *The assumption is actually general. Regarding Λ as an action space and $r_\lambda = \mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda^{(t)})$, when solving $\arg\max_{\mathcal{P}_\Lambda} \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda}[\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda^{(t)})] = \arg\max_{\mathcal{P}_\Lambda} \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda}[r_\lambda]$, the data distribution optimization operator $\mathcal{E}$ is equivalent to solving a multi-arm bandit (MAB) problem. For the first-order optimization, (Schulman et al., 2017a) shows that the solution of a KL-regularized version, $\arg\max_{\mathcal{P}_\Lambda} \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda}[r_\lambda] - \eta KL(\mathcal{P}_\Lambda||\mathcal{P}_\Lambda^{(t)})$, is exactly the assumption. For the second-order optimization, let $\mathcal{P}_\Lambda = softmax(\{r_\lambda\})$, (Agarwal et al., 2019) shows that the natural policy gradient of a softmax parameterization also induces exactly the assumption.*

Assumption 3 (First-Order Optimization Co-Monotonic Assumption)**.** *For $\forall\, \lambda_1, \lambda_2 \in \Lambda$, we have $[\mathcal{L}_\mathcal{E}(\lambda_1, \theta_{\lambda_1}) - \mathcal{L}_\mathcal{E}(\lambda_2, \theta_{\lambda_2})] \cdot [\mathcal{L}_\mathcal{T}(\lambda_1, \theta_{\lambda_1}) - \mathcal{L}_\mathcal{T}(\lambda_2, \theta_{\lambda_2})] \geq 0$.*

Assumption 4 (Second-Order Optimization Co-Monotonic Assumption)**.** *For $\forall\, \lambda_1, \lambda_2 \in \Lambda$, $\exists\, \eta_0 > 0$, s.t. $\forall\, 0 < \eta < \eta_0$, we have $[\mathcal{L}_\mathcal{E}(\lambda_1, \theta_{\lambda_1}) - \mathcal{L}_\mathcal{E}(\lambda_2, \theta_{\lambda_2})] \cdot [G^\eta\mathcal{L}_\mathcal{T}(\lambda_1, \theta_{\lambda_1}) - G^\eta\mathcal{L}_\mathcal{T}(\lambda_2, \theta_{\lambda_2})] \geq 0$, where $\theta_\lambda^\eta = \theta_\lambda + \eta\nabla_{\theta_\lambda}\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)$ and $G^\eta\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda) = \frac{1}{\eta}[\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^\eta) - \mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)]$.*

Under assumptions (1) (2) (3), if $\mathcal{T}$ is a first-order operator, namely a gradient accent operator, to maximize $\mathcal{L}_\mathcal{T}$, GDI can be guaranteed to be superior to that w/o $\mathcal{E}$. Under assumptions (1) (2) (4), if $\mathcal{T}$ is a second-order operator, namely a natural gradient operator, to maximize $\mathcal{L}_\mathcal{T}$, GDI can also be guaranteed to be superior to that w/o $\mathcal{E}$.

Theorem 1 (First-Order Optimization with Superior Target)**.** *Under assumptions (1) (2) (3), we have $\mathcal{L}_\mathcal{T}(\mathcal{P}_\Lambda^{(t+1)}, \theta^{(t+1)}) = \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda^{(t+1)}}[\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1)})] \geq \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}[\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1)})] = \mathcal{L}_\mathcal{T}(\mathcal{P}_\Lambda^{(t)}, \theta^{(t+1)})$.*

Proof. *By **Theorem 4** (see App. C), the upper triangular transport inequality, let $f(\lambda) = \mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)$ and $g(\lambda) = \mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda)$, the proof is done.*

Remark (Superiority of Target)**.** *In Algorithm 1, if $\mathcal{E}$ updates $\mathcal{P}_\Lambda^{(t)}$ at time t, then the operator $\mathcal{T}$ at time $t+1$ can be written as $\theta^{(t+2)} = \theta^{(t+1)} + \eta\nabla_{\theta^{(t+1)}}\mathcal{L}_\mathcal{T}(\mathcal{P}_\Lambda^{(t+1)}, \theta^{(t+1)})$. If $\mathcal{P}_\Lambda^{(t)}$ hasn't been updated at time t, then the operator $\mathcal{T}$ at time $t+1$ can be written as $\theta^{(t+2)} = \theta^{(t+1)} + \eta\nabla_{\theta^{(t+1)}}\mathcal{L}_\mathcal{T}(\mathcal{P}_\Lambda^{(t)}, \theta^{(t+1)})$. **Theorem 1** shows that the target of $\mathcal{T}$ at time $t+1$ becomes higher if $\mathcal{P}_\Lambda^{(t)}$ is updated by $\mathcal{E}$ at time t.*

Example 1 (Practical Implementation)**.** *Let $\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda) = \mathcal{J}_{\pi_{\theta_\lambda}}$ and $\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda) = \mathcal{J}_{\pi_{\theta_\lambda}}$. $\mathcal{E}$ can update $\mathcal{P}_\Lambda$ by the Monte-Carlo estimation of $\mathcal{J}_{\pi_{\theta_\lambda}}$. $\mathcal{T}$ is to maximize $\mathcal{J}_{\pi_{\theta_\lambda}}$, which can be any RL algorithms.*

Theorem 2 (Second-Order Optimization with Superior Improvement)**.** *Under assumptions (1) (2) (4), we have $\boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda^{(t+1)}}[G^\eta\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1)})] \geq \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}[G^\eta\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1)})]$, more specifically,*

$$\begin{aligned}&\boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda^{(t+1)}}[\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1),\eta}) - \mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1)})]\\ &\geq \boldsymbol{E}_{\lambda\sim\mathcal{P}_\Lambda^{(t)}}[\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1),\eta}) - \mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda^{(t+1)})]\end{aligned}$$

Table 1. Experiment results of Atari. Playtime is the equivalent human playtime, HWRB is the human world record breakthrough, HNS is the human normalized score, HWRNS is the human world records normalized score, SABER $= \max\{\min\{\text{HWRNS}, 2\}, 0\}$.

	GDI-H^3	GDI-I^3	Muesli	RAINBOW	LASER	R2D2	NGU	Agent57
Training Scale (Num. Frames)	**2E+8**	**2E+8**	**2E+8**	**2E+8**	**2E+8**	1E+10	3.5E+10	1E+11
Playtime (Day)	**38.5**	**38.5**	**38.5**	**38.5**	**38.5**	1929	6751.5	19290
HWRB	**22**	17	5	4	7	15	8	18
Mean HNS(%)	**9620.33**	7810.1	2538.12	873.54	1740.94	3373.48	3169.07	4762.17
Median HNS(%)	1146.39	832.5	1077.47	230.99	454.91	1342.27	1174.92	**1933.49**
Mean HWRNS(%)	**154.27**	117.98	75.52	28.39	45.39	98.78	76.00	125.92
Median HWRNS(%)	**50.63**	35.78	24.86	4.92	8.08	33.62	21.19	43.62
Mean SABER(%)	71.26	61.66	48.74	28.39	36.78	60.43	50.47	**76.26**
Median SABER(%)	**50.63**	35.78	24.86	4.92	8.08	33.62	21.19	43.62

Proof. *By **Theorem** 4 (see App. C), the upper triangular transport inequality, let $f(\lambda) = G^\eta \mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)$ and $g(\lambda) = \mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda)$, the proof is done.*

Remark (Superiority of Improvement)**.** ***Theorem** 2 shows that, if $\mathcal{P}_\Lambda$ is updated by $\mathcal{E}$, the expected improvement of $\mathcal{T}$ is higher.*

Example 2 (Practical Implementation)**.** *Let $\mathcal{L}_\mathcal{E}(\lambda, \theta_\lambda) = \boldsymbol{E}_{s\sim d^\pi_{\rho_0}} \boldsymbol{E}_{a\sim\pi(\cdot|s)\exp(\epsilon A^\pi(s,\cdot))/Z}[A^\pi(s,a)]$, where $\pi = \pi_{\theta_\lambda}$. Let $\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda) = \mathcal{J}_{\pi_{\theta_\lambda}}$. If we optimize $\mathcal{L}_\mathcal{T}(\lambda, \theta_\lambda)$ by natural gradient, (Agarwal et al., 2019) shows that, for direct parameterization, the natural policy gradient gives $\pi^{(t+1)} \propto \pi^{(t)} \exp(\epsilon A^{\pi^{(t)}})$, by **Lemma** 4 (see App. C), the performance difference lemma, $V^\pi(s_0) - V^{\pi'}(s_0) = \frac{1}{1-\gamma} \boldsymbol{E}_{s\sim d^\pi_{s_0}} \boldsymbol{E}_{a\sim\pi(\cdot|s)}[A^{\pi'}(s,a)]$, hence if we ignore the gap between the states visitation distributions of $\pi^{(t)}$ and $\pi^{(t+1)}$, $\mathcal{L}_\mathcal{E}(\lambda, \theta^{(t)}_\lambda) \approx \frac{1}{1-\gamma} \boldsymbol{E}_{s\sim d^\pi_{\rho_0}}[V^{\pi^{(t+1)}}(s) - V^{\pi^{(t)}}(s)]$, where $\pi^{(t)} = \pi_{\theta^{(t)}_\lambda}$. Hence, $\mathcal{E}$ is actually putting more measure on λ that can achieve more improvement.*

5. Experiment

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

- How to implement RL algorithms based on GDI step by step (see Sec. 5.1)? Whether the proposed methods can outperform all prior SOTA RL algorithms in both sample efficiency and final performance (see Tab. 1)?
- How to construct a behavior policy space (see Sec. 5.1)? What's the impact of the size of the index set Λ, namely, whether the data richness can be improved via increasing the capacity and diversity (see Fig. 3)?
- How to design a data distribution optimization operator (e.g., a meta-controller) to tackle the exploration and exploitation trade-off (see Sec. 5.1)? How much performance would be degraded without data distribution optimization, namely no meta-controller (see Fig. 3)?

5.1. Practical Implementation Based on GDI

Policy Space Construction To illustrate the effectiveness of GDI, we give two representative practical implementations of GDI, namely GDI-I^3 and GDI-H^3, the capacity of whose behavior policy space is larger than Agent57. Let $\Lambda = \{\lambda|\lambda = (\tau_1, \tau_2, \epsilon)\}$. The behavior policy belongs to a *soft* entropy policy space including policies ranging from very exploratory to purely exploitative and thereby the optimization of the sampling distribution of behavior policy $\mathcal{P}_\Lambda$ can be reframed into the trade-off between exploration and exploitation. We define the behavior policy π_{θ_λ} as

$$\pi_{\theta_\lambda} = \epsilon\cdot\text{Softmax}\left(\frac{A_{\theta_1}}{\tau_1}\right) + (1-\epsilon)\cdot\text{Softmax}\left(\frac{A_{\theta_2}}{\tau_2}\right) \quad (2)$$

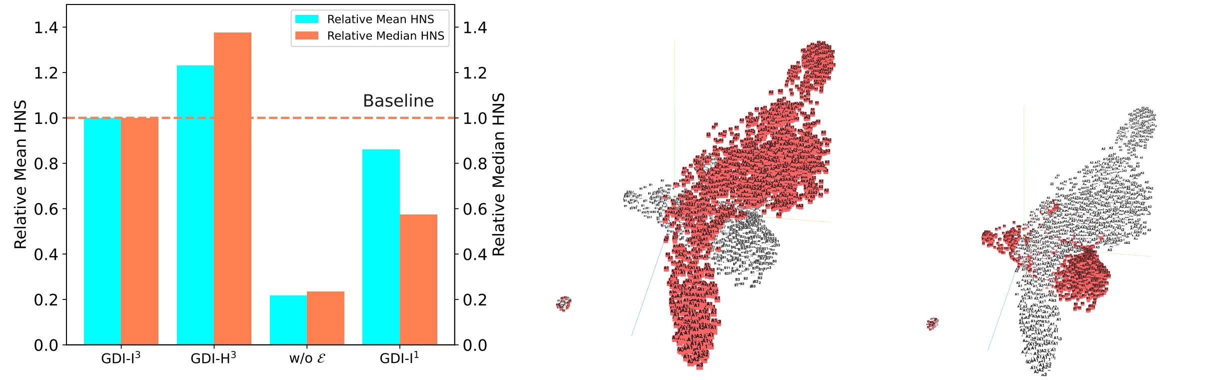
wherein π_{θ_λ} constructs a parameterized policy space, and the index set Λ is constructed by $\lambda = (\tau_1, \tau_2, \epsilon)$. For GDI-I^3, A_{θ_1} and A_{θ_2} are identical advantage functions (Wang et al., 2016). Namely, they are estimated by an isomorphic family of trainable variables θ. The learning policy is also π_{θ_λ}. For GDI-H^3, A_{θ_1} and A_{θ_2} are different, and they are estimated by two different families of trainable variables (i.e., $\theta_1 \neq \theta_2$). Since **GDI needn't assume** A_{θ_1} **and** A_{θ_2} **are learned from the same MDP**, we adopt two kinds of reward shaping to learn A_{θ_1} and A_{θ_2} respectively, which can see App. G. More implementation details see App. D.

Data Distribution Optimization Operator The operator $\mathcal{E}$, which optimizes $\mathcal{P}_\Lambda$, is achieved by Multi-Arm Bandits (Sutton & Barto, 2018, MAB), where assumption (2) holds naturally. For more details, can see App. E.

Reinforcement Learning Optimization Operator The operator $\mathcal{T}$ is achieved by policy gradient, V-Trace and ReTrace (Espeholt et al., 2018; Munos et al., 2016) (see App. B), which meets Theorem 1 by first-order optimization.

5.2. Summary of Results

Experimental Details Recommended by (Badia et al., 2020a; Toromanoff et al., 2019), we construct an evaluation



(a) Performance of Control Groups (b) t-SNE of GDI-I^3 (c) t-SNE of GDI-I^1

Figure 3. Figures of ablation study. **(a)** shows how the ablation groups (see App. L) perform compared with the baseline (i.e., GDI-I^3). Noting that the performance has been normalized by GDI-I^3 (e.g., $\frac{\text{Mean HNS of GDI-I}^1}{\text{Mean HNS of GDI-I}^3}$), and w/o $\mathcal{E}$ means without the meta-controller. **(b)** and **(c)** illustrate the data richness (e.g., $\frac{\text{Seen Conditions}}{\text{All Conditions}}$) of GDI-I^1 and GDI-I^3 via t-SNE of visited states (see App. L.2).

system to highlight the superiority of GDI from multiple levels (see App. H). Furthermore, to avoid any issues that aggregated metrics may have, App. K provides full learning curves for all games and detailed comparison tables of raw and normalized scores. More details see App. F.

Effectiveness of GDI The aggregated results across games are reported in Tab. 1. Our agents obtain the highest mean HNS with the minimal training frames, leading to the best learning efficiency. Furthermore, our agents have surpassed 22 human world records within 38 playtime days, which is **500 times** more efficient than Agent57. Extensive experiments have demonstrated the fact that either GDI-I^3 or GDI-H^3 could obtain superhuman performance with remarkable learning efficiency.

Discussion of the Results Agent57 could obtain the highest median HNS but relatively lower learning efficiency via **i)** a relatively larger behavior policy space and a meta-controller **ii)** intrinsic rewards and nearly unlimited data. However, Agent57 fails to distinguish the value of data and thereby collects many useless/low-value samples. Other algorithms are struggling to match our performance.

5.3. Ablation Study

Ablation Study Design In the ablation study, we further investigate the effects of several properties of GDI. In the first experiment, we demonstrate the effectiveness of the capacity and diversity control via exploring how different sizes of the index set of the policy space influence the performance and data richness. In the second experiment, we highlight the effectiveness of data distribution optimization operator $\mathcal{E}$ via ablating $\mathcal{E}$. More details can see App. L.

Effectiveness of Capacity and Diversity Control In this experiment, we firstly implement a GDI-I^1 algorithm with Boltzmann policy space (i.e., $\pi_{\theta_\lambda} = \text{Softmax}(\frac{A}{\tau})$) to explore the impact of the capacity and diversity control. Then, we explore whether the data richness is indeed improved via a case study of t-SNE of GDI-I^3 and GDI-I^1. Results are illustrated in Fig. 3, from which we could find the visited states of GDI-I^3 are indeed richer than GDI-I^1, which concludes its better performance. In the same way, the behavior policy space of GDI-I^3 is a sub-space (i.e., $\theta_1 = \theta_2$) of that of GDI-H^3, leading to further performance improvement.

Effectiveness of Data Distribution Optimization From Fig. 3, we could also find that not using a meta-controller (e.g., the index λ of behavior policy takes a fixed value) will dramatically degrade performance, which confirms the effectiveness of the data distribution optimization and echoes the previous theoretical proof.

6. Conclusion

Simultaneously obtaining superior sample efficiency and better final performance is an important and challenging problem in RL. In this paper, we present the first attempt to address this problem from training data distribution control, namely to obtain any desired (e.g., nontrivial) data within *limited* interactions. To tackle this problem, we firstly cast it into a data distribution optimization problem. Then, we handle this problem via **i)** explicitly modeling and controlling the diversity of the behavior policies and **ii)** adaptively tackling the exploration-exploitation trade-off using meta-learning. After integrating this process into GPI, we surprisingly find a more general framework GDI and then we give an operation-version of recent SOTA algorithms. Under the guidance of GDI, we propose feasible implementations and achieve the superhuman final performance with remarkable learning efficiency within only 38 playtime days.

Acknowledgements

We are grateful for the careful reading and insightful reviews of meta-reviewers and reviewers.

References

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *arXiv preprint arXiv:1908.00261*, 2019.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., and Blundell, C. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020a.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020b.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Dadashi, R., Taiga, A. A., Le Roux, N., Schuurmans, D., and Bellemare, M. G. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, pp. 1486–1495. PMLR, 2019.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Fan, J. A review for deep reinforcement learning in atari: Benchmarks, challenges, and solutions. *CoRR*, abs/2112.04145, 2021. URL https://arxiv.org/abs/2112.04145.

Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R. P., and Levine, S. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in Neural Information Processing Systems*, 34, 2021.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

Hessel, M., Danihelka, I., Viola, F., Guez, A., Schmitt, S., Sifre, L., Weber, T., Silver, D., and van Hasselt, H. Muesli: Combining improvements in policy optimization. *arXiv preprint arXiv:2104.06159*, 2021.

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

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Kumar, A., Gupta, A., and Levine, S. Discor: Corrective feedback in reinforcement learning via distribution correction. *arXiv preprint arXiv:2003.07305*, 2020.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M. J., and Bowling, M. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 1054–1062. Curran Associates, Inc., 2016.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017. URL http://arxiv.org/abs/1705.05363.

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

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Schmidhuber, S. H. J. Long short-term memory. *Neural Computation.*, 1997.

Schmitt, S., Hessel, M., and Simonyan, K. Off-policy actor-critic with shared experience replay. In *International Conference on Machine Learning*, pp. 8545–8554. PMLR, 2020.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Schulman, J., Chen, X., and Abbeel, P. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Toromanoff, M., Wirbel, E., and Moutarde, F. Is deep reinforcement learning really superhuman on atari? leveling the playing field. *arXiv preprint arXiv:1908.04683*, 2019.

Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B., and Gershman, S. J. Human learning in atari. In *2017 AAAI Spring Symposium Series*, 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003, 2016.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Watkins, C. J. C. H. *Learning from delayed rewards*. King's College, Cambridge United Kingdom, 1989.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Xiao, C., Shi, H., Fan, J., and Deng, S. CASA: A bridge between gradient of policy improvement and policy evaluation. *CoRR*, abs/2105.03923, 2021a. URL https://arxiv.org/abs/2105.03923.

Xiao, C., Shi, H., Fan, J., and Deng, S. An entropy regularization free mechanism for policy-based reinforcement learning. *CoRR*, abs/2106.00707, 2021b. URL https://arxiv.org/abs/2106.00707.

A. Summary of Notation and Abbreviation

In this section, we briefly summarize some common notations and abbreviations in this paper for the convenience of readers, which are illustrated in Tab. 2 and Tab. 3.

Table 2. Summary of Notation

Notation	Description
s	state
a	action
$\mathcal{S}$	set of all states
$\mathcal{A}$	set of all actions
Δ	probability simplex
μ	behavior policy
π	target policy
G_t	cumulative discounted reward or return at t
$d_{\rho_0}^{\pi}$	the states visitation distribution of π with the initial state distribution ρ_0
J_π	the expectation of the returns with the states visitation distribution of π
V^π	the state value function of π
Q^π	the state-action value function of π
γ	discount-rate parameter
δ_t	temporal-difference error at t
Λ	set of indexes
λ	one index in Λ
$\mathcal{P}_\Lambda$	one probability measure on Λ
Θ	set of all possible parameter values
θ	one parameter value in Θ
θ_λ	a subset of θ, indicates the parameter in θ being used by the index λ
$\mathcal{X}$	set of samples
x	one sample in $\mathcal{X}$
$\mathcal{D}$	set of all possible states visitation distributions
$\mathcal{E}$	the data distribution optimization operator
$\mathcal{T}$	the RL algorithm optimization operator
$L_{\mathcal{E}}$	the loss function of $\mathcal{E}$ to be maximized, calculated by the samples set $\mathcal{X}$
$\mathcal{L}_{\mathcal{E}}$	expectation of $L_{\mathcal{E}}$, with respect to each sample $x \in \mathcal{X}$
$L_{\mathcal{T}}$	the loss function of $\mathcal{T}$ to be maximized, calculated by the samples set $\mathcal{X}$
$\mathcal{L}_{\mathcal{T}}$	expectation of $L_{\mathcal{T}}$, with respect to each sample $x \in \mathcal{X}$

Table 3. Summary ofAbbreviation

Abbreviation	**Description**
Sec.	Section (Badia et al., 2020a)
Figs.	Figures (Hafner et al., 2020)
Fig.	Figure (Badia et al., 2020a)
Eq.	Equation (Badia et al., 2020a)
Tab.	Table (Badia et al., 2020a)
App.	Appendix (Badia et al., 2020a)
SOTA	State-of-The-Art (Badia et al., 2020a)
RL	Reinforcement Learning (Sutton & Barto, 2018)
DRL	Deep Reinforcement Learning (Sutton & Barto, 2018)
GPI	Generalized Policy Iteration (Sutton & Barto, 2018)
PG	Policy Gradient (Sutton & Barto, 2018)
AC	Actor Critic (Sutton & Barto, 2018)
ALE	Atari Learning Environment (Bellemare et al., 2013)
HNS	Human Normalized Score (Bellemare et al., 2013)
HWRB	Human World Records Breakthrough
HWRNS	Human World Records Normalized Score
SABER	Standardized Atari BEnchmark for RL (Toromanoff et al., 2019)
CHWRNS	Capped Human World Records Normalized Score
WLOG	Without Loss of Generality
w/o	Without

B. Background on RL

The RL problem can be formulated by a Markov decision process (Howard, 1960, MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$. Considering a discounted episodic MDP, the initial state s_0 will be sampled from the distribution denoted by $\rho_0(s) : \mathcal{S} \rightarrow \Delta(\mathcal{S})$. At each time t, the agent choose an action $a_t \in \mathcal{A}$ according to the policy $\pi(a_t|s_t) : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ at state $s_t \in \mathcal{S}$. The environment receives the action, produces a reward $r_t \sim r(s,a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbf{R}$ and transfers to the next state s_{t+1} submitted to the transition distribution $p(s' \mid s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$. The process continues until the agent reaches a terminal state or a maximum time step. Define return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, state value function $V^\pi(s_t) = \mathbf{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t\right]$, state-action value function $Q^\pi(s_t, a_t) = \mathbf{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t, a_t\right]$, and advantage function $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$, wherein $\gamma \in (0, 1)$ is the discount factor. The connections between V^π and Q^π is given by the Bellman equation,

$$\mathcal{T} Q^\pi(s_t, a_t) = \mathbf{E}_{'\pi}[r_t + \gamma V^\pi(s_{t+1})],$$

where

$$V^\pi(s_t) = \mathbf{E}_\pi[Q^\pi(s_t, a_t)].$$

The goal of reinforcement learning is to find the optimal policy π^* that maximizes the expected sum of discounted rewards, denoted by $\mathcal{J}$ (Sutton & Barto, 2018):

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathcal{J}_\pi(\tau) = \underset{\pi}{\operatorname{argmax}} \mathbf{E}_\pi\left[G_t\right] = \underset{\pi}{\operatorname{argmax}} \mathbf{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k}]$$

Model-free reinforcement learning (MFRL) has made many impressive breakthroughs in a wide range of Markov decision processes (Vinyals et al., 2019; Pedersen, 2019; Badia et al., 2020a, MDP). MFRL mainly consists of two categories, valued-based methods (Mnih et al., 2015; Hessel et al., 2017) and policy-based methods (Schulman et al., 2015; 2017b; Espeholt et al., 2018).

Value-based methods learn state-action values and select actions according to these values. One merit of value-based methods is to accurately control the exploration rate of the behavior policies by some trivial mechanism, such like ϵ-greedy. The drawback is also apparent. The policy improvement of valued-based methods totally depends on the policy evaluation. Unless the selected action is changed by a more accurate policy evaluation, the policy won't be improved. So the policy improvement of each policy iteration is limited, which leads to a low learning efficiency. Previous works equip valued-based methods with many appropriated designed structures, achieving a more promising learning efficiency (Wang et al., 2016; Schaul et al., 2015; Kapturowski et al., 2018).

In practice, value-based methods maximize $\mathcal{J}$ by policy iteration (Sutton & Barto, 2018). The policy evaluation is fulfilled by minimizing $\mathbf{E}_\pi[(G - Q^\pi)^2]$, which gives the gradient ascent direction $\mathbf{E}_\pi[(G - Q^\pi)\nabla Q^\pi]$. The policy improvement is usually achieved by ϵ-greedy.

Q-learning is a typical value-based methods, which updates the state-action value function $Q(s, a)$ with Bellman Optimality Equation (Watkins & Dayan, 1992):

$$\begin{gathered}\delta_t = r_{t+1} + \gamma \arg\max_a Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right) \\ Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha \delta_t\end{gathered}$$

wherein δ_t is the temporal difference error (Sutton, 1988), and α is the learning rate.

A refined structure design of Q^π is achieved by (Wang et al., 2016). It estimates Q^π by a summation of two separated networks, $Q^\pi = A^\pi + V^\pi$, which has been widely studied in (Wang et al., 2016; Xiao et al., 2021a).

Policy gradient (Williams, 1992, PG) methods is an outstanding representative of policy-based RL algorithms, which directly parameterizes the policy and updates through optimizing the following objective:

$$\mathcal{J}(\theta) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \log \pi_\theta\left(a_t \mid s_t\right) R(\tau)\right]$$

wherein $R(\tau)$ is the cumulative return on trajectory τ. In PG method, policy improves via ascending along the gradient of the above equation, denoted as policy gradient:

$$\nabla_\theta \mathcal{J}\left(\pi_\theta\right)=\underset{\tau \sim \pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) R(\tau)\right]$$

One merit of policy-based methods is that they incorporate a policy improvement phase every training step, suggesting a higher learning efficiency than value-based methods. Nevertheless, policy-based methods easily fall into a suboptimal solution, where the entropy drops to 0 (Haarnoja et al., 2018). The actor-critic methods introduce a value function as the baseline to reduce the variance of the policy gradient (Mnih et al., 2016), but maintain the other characteristics unchanged.

Actor-Critic (Sutton & Barto, 2018, AC) reinforcement learning updates the policy gradient with an value-based critic, which can reduce variance of estimates and thereby ensure more stable and rapid optimization.

$$\nabla_\theta \mathcal{J}(\theta)=\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \psi_t \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$$

wherein ψ_t is the critic to guide the improvement directions of policy improvement, which can be the state-action value function $Q^\pi\left(s_t, a_t\right)$, the advantage function $A^\pi\left(s_t, a_t\right)=Q^\pi\left(s_t, a_t\right)-V^\pi(s_t)$.

B.1. Retrace

When large scale training is involved, the off-policy problem is inevitable. Denote μ to be the behavior policy, π to be the target policy, and $c_t=\min\{\frac{\pi_t}{\mu_t}, \bar{c}\}$ to be the clipped importance sampling. For brevity, denote $c_{[t:t+k]}=\prod_{i=0}^{k} c_{t+i}$. ReTrace (Munos et al., 2016) estimates $Q(s_t, a_t)$ by clipped per-step importance sampling

$$Q^{\tilde{\pi}}(s_t, a_t)=\mathbf{E}_\mu[Q(s_t, a_t)+\sum_{k \geq 0} \gamma^k c_{[t+1:t+k]} \delta_{t+k}^Q Q],$$

where $\delta_t^Q Q \overset{def}{=} r_t+\gamma Q(s_{t+1}, a_{t+1})-Q(s_t, a_t)$. The above operator is a contraction mapping, and Q converges to $Q^{\tilde{\pi}_{ReTrace}}$ that corresponds to some $\tilde{\pi}_{ReTrace}$.

B.2. Vtrace

Policy-based methods maximize $\mathcal{J}$ by policy gradient. It's shown (Sutton & Barto, 2018) that $\nabla \mathcal{J}=\mathbf{E}_\pi[G \nabla \log \pi]$. When involved with a baseline, it becomes an actor-critic algorithm such as $\nabla \mathcal{J}=\mathbf{E}_\pi[(G-V^\pi) \nabla \log \pi]$, where V^π is optimized by minimizing $\mathbf{E}_\pi[(G-V^\pi)^2]$, i.e. gradient ascent direction $\mathbf{E}_\pi[(G-V^\pi) \nabla V^\pi]$.

IMPALA (Espeholt et al., 2018) introduces V-Trace off-policy actor-critic algorithm to correct for the discrepancy between target policy and behavior policy. Denote $\rho_t=\min\{\frac{\pi_t}{\mu_t}, \bar{\rho}\}$. V-Trace estimates $V(s_t)$ by

$$V^{\tilde{\pi}}(s_t)=\mathbf{E}_\mu[V(s_t)+\sum_{k \geq 0} \gamma^k c_{[t:t+k-1]} \rho_{t+k} \delta_{t+k}^V V],$$

where $\delta_t^V V \overset{def}{=} r_t+\gamma V(s_{t+1})-V(s_t)$. If $\bar{c} \leq \bar{\rho}$, the above operator is a contraction mapping, and V converges to $V^{\tilde{\pi}}$ that corresponds to

$$\tilde{\pi}(a|s)=\frac{\min\left\{\bar{\rho} \mu(a|s), \pi(a|s)\right\}}{\sum_{b \in \mathcal{A}} \min\left\{\bar{\rho} \mu(b|s), \pi(b|s)\right\}}.$$

The policy gradient is given by

$$\mathbf{E}_\mu\left[\rho_t(r_t+\gamma V^{\tilde{\pi}}(s_{t+1})-V(s_t)) \nabla \log \pi\right].$$

C. Theoretical Proof

For a monotonic sequence of numbers which satisfies $a = x_0 < x_1 < \cdots < x_n < b$, we call it a split of interval $[a, b]$.

Lemma 1 (Discretized Upper Triangular Transport Inequality for Increasing Functions in $\mathbf{R}^1$)**.** *Assume μ is a continuous probability measure supported on $[0,1]$. Let $0 = x_0 < x_1 < \cdots < x_n < 1$ to be any split of $[0,1]$. Define $\tilde{\mu}(x_i) = \mu([x_i, x_{i+1}))$. Define*

$$\tilde{\beta}(x_i) = \tilde{\mu}(x_i)\exp(x_i)/Z,\ Z = \sum_i \tilde{\mu}(x_i)\exp(x_i).$$

Then there exists a probability measure $\gamma : \{x_i\}_{i=0,\dots,n} \times \{x_i\}_{i=0,\dots,n} \to [0,1]$, s.t.

$$\begin{cases} \sum_j \gamma(x_i, y_j) = \tilde{\mu}(x_i), & i = 0, \dots, n; \\ \sum_i \gamma(x_i, y_j) = \tilde{\beta}(y_j), & j = 0, \dots, n; \\ \gamma(x_i, y_j) = 0, & i > j. \end{cases} \tag{3}$$

Then for any monotonic increasing function $f : \{x_i\}_{i=0,\dots,n} \to \boldsymbol{R}$, we have

$$\boldsymbol{E}_{\tilde{\mu}}[f] \le \boldsymbol{E}_{\tilde{\beta}}[f].$$

Proof of Lemma 1. For any couple of measures (μ, β), we say the couple satisfies Upper Triangular Transport Condition (UTTC), if there exists γ s.t. (3) holds.

Given $0 = x_0 < x_1 < \cdots < x_n < 1$, we prove the existence of γ by induction.

Define

$$\tilde{\mu}_m(x_i) = \begin{cases} \mu([x_i, x_{i+1})), & i < m, \\ \mu([x_i, 1)), & i = m, \\ 0, & i > m. \end{cases}$$

Define

$$\tilde{\beta}_m(x_i) = \tilde{\mu}_m(x_i)\exp(x_i)/Z_m,\ Z_m = \sum_i \tilde{\mu}_m(x_i)\exp(x_i).$$

Noting if we prove that $(\tilde{\mu}_m, \tilde{\beta}_m)$ satisfies UTTC for $m = n$, it's equivalent to prove the existence of γ in (3).

To clarify the proof, we use x_i to represent the point for $\tilde{\mu}$-axis in coupling and y_j to represent the point for $\tilde{\beta}$-axis, but they are actually identical, i.e. $x_i = y_j$ when $i = j$.

When $m = 0$, it's obvious that $(\tilde{\mu}_0, \tilde{\beta}_0)$ satisfies UTTC, as

$$\gamma_0(x_i, y_j) = \begin{cases} 1,\ i = 0, j = 0, \\ 0,\ else. \end{cases}$$

Assume UTTC holds for m, i.e. there exists γ_m s.t. $(\tilde{\mu}_m, \tilde{\beta}_m)$ satisfies UTTC, we want to prove it also holds for $m+1$.

By definition of $\tilde{\mu}_m$, we have

$$\begin{cases} \tilde{\mu}_m(x_i) = \tilde{\mu}_{m+1}(x_i), & i < m, \\ \tilde{\mu}_m(x_i) = \tilde{\mu}_{m+1}(x_i) + \tilde{\mu}_{m+1}(x_{i+1}), & i = m, \\ \tilde{\mu}_m(x_{m+1}) = \tilde{\mu}_m(x_i) = \tilde{\mu}_{m+1}(x_i) = 0, & i > m+1. \end{cases}$$

By definition of $\tilde{\beta}_m$, we have

$$\begin{cases} \tilde{\beta}_m(x_i) = \tilde{\beta}_{m+1}(x_i) \cdot \dfrac{Z_{m+1}}{Z_m}, & i < m, \\ \tilde{\beta}_m(x_i) = \Big(\tilde{\beta}_{m+1}(x_i) + \tilde{\beta}_{m+1}(x_{i+1}) \exp(x_i - x_{i+1})\Big) \cdot \dfrac{Z_{m+1}}{Z_m}, & i = m, \\ \tilde{\beta}_m(x_{m+1}) = \tilde{\beta}_m(x_i) = \tilde{\beta}_{m+1}(x_i) = 0, & i > m+1. \end{cases}$$

Multiplying γ_m by $\frac{Z_m}{Z_{m+1}}$, we get the following UTTC

$$\begin{cases} \sum_j \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = \dfrac{Z_m}{Z_{m+1}} \tilde{\mu}_{m+1}(x_i), & i < m; \\ \sum_j \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = \dfrac{Z_m}{Z_{m+1}} (\tilde{\mu}_{m+1}(x_i) + \tilde{\mu}_{m+1}(x_{i+1})), & i = m; \\ \sum_j \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = 0, & i = m+1; \\ \sum_j \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = \tilde{\mu}_{m+1}(x_i) = 0, & i > m+1; \\ \sum_i \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = \tilde{\beta}_{m+1}(y_j), & j < m; \\ \sum_i \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = \tilde{\beta}_{m+1}(y_i) + \tilde{\beta}_{m+1}(y_{j+1}) \exp(y_j - y_{j+1}), & j = m; \\ \sum_i \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = 0, & j = m+1; \\ \sum_i \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = \tilde{\beta}_{m+1}(y_j) = 0, & j > m+1; \\ \dfrac{Z_m}{Z_{m+1}} \gamma_m(x_i, y_j) = 0, & i > j. \end{cases}$$

By definition of Z_m,

$$Z_{m+1} - Z_m = \tilde{\mu}_{m+1}(x_{m+1})(\exp(x_{m+1}) - \exp(x_m)) > 0, \tag{4}$$

so we have $\frac{Z_m}{Z_{m+1}} \tilde{\mu}_{m+1}(x_i) < \tilde{\mu}_{m+1}(x_i)$.

Noticing that $\tilde{\beta}_{m+1}(y_{i+1}) \exp(y_i - y_{i+1}) < \tilde{\beta}_{m+1}(y_{i+1})$ and $\frac{Z_m}{Z_{m+1}} \tilde{\mu}_{m+1}(x_i) < \tilde{\mu}_{m+1}(x_i)$, we decompose the measure of $\frac{Z_m}{Z_{m+1}} \gamma_m$ at (x_i, y_m) to $(x_i, y_m), (x_i, y_{m+1})$ for $i = 0, \ldots, m-1$, and complement a positive measure at (x_i, y_{m+1}) to make up the difference between $\frac{Z_m}{Z_{m+1}} \tilde{\mu}_{m+1}(x_i)$ and $\tilde{\mu}_{m+1}(x_i)$. For $i = m$, we decompose the measure at (x_m, y_m) to $(x_m, y_m), (x_m, y_{m+1}), (x_{m+1}, y_{m+1})$ and also complement a proper positive measure.

Now we define γ_{m+1} by

$$\begin{cases} \gamma_{m+1}(x_i, y_j) = \dfrac{Z_m}{Z_{m+1}}\gamma_m(x_i, y_j), & i < m \textit{ and } j < m, \\ \gamma_{m+1}(x_i, y_j) = \left(\dfrac{Z_m}{Z_{m+1}}\gamma_m(x_i, y_j) + \dfrac{Z_{m+1} - Z_m}{Z_{m+1}}\tilde{\mu}_{m+1}(x_i)\right) & \\ \qquad\qquad \cdot \dfrac{\tilde{\beta}_{m+1}(y_j)}{\tilde{\beta}_{m+1}(y_j) + \tilde{\beta}_{m+1}(y_{j+1})}, & i < m \textit{ and } j = m, \\ \gamma_{m+1}(x_i, y_j) = \left(\dfrac{Z_m}{Z_{m+1}}\gamma_m(x_i, y_j) + \dfrac{Z_{m+1} - Z_m}{Z_{m+1}}\tilde{\mu}_{m+1}(x_i)\right) & \\ \qquad\qquad \cdot \dfrac{\tilde{\beta}_{m+1}(y_{j+1})}{\tilde{\beta}_{m+1}(y_j) + \tilde{\beta}_{m+1}(y_{j+1})}, & i < m \textit{ and } j = m+1, \\ \gamma_{m+1}(x_i, y_j) = 0, & i > j \textit{ or } i > m+1 \textit{ or } j > m+1, \\ \gamma_{m+1}(x_m, y_m) = u, & \\ \gamma_{m+1}(x_m, y_{m+1}) = v, & \\ \gamma_{m+1}(x_{m+1}, y_{m+1}) = w, & \end{cases}$$

where we assume u, v, w to be the solution of the following equations

$$\begin{cases} u + v + w = \tilde{\mu}_{m+1}(x_m) + \tilde{\mu}_{m+1}(x_{m+1}), \\ \dfrac{w}{u+v} = \dfrac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)}, \\ \dfrac{v+w}{u} = \dfrac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)}, \\ u, v, w \geq 0. \end{cases} \tag{5}$$

It's obvious that

$$\begin{cases} \displaystyle\sum_j \gamma_{m+1}(x_i, y_j) = \tilde{\mu}_{m+1}(x_i) = 0, & i > m+1, \\ \displaystyle\sum_i \gamma_{m+1}(x_i, y_j) = \tilde{\beta}_{m+1}(y_j) = 0, & j > m+1, \\ \gamma(x_i, y_j) = 0, & i > j. \end{cases}$$

For $j < m$, since $\sum_i \frac{Z_m}{Z_{m+1}}\gamma_m(x_i, y_j) = \tilde{\beta}_{m+1}(y_j)$, we have

$$\sum_i \gamma_{m+1}(x_i, y_j) = \tilde{\beta}_{m+1}(y_j),\ j < m.$$

For $i < m$, since $\sum_j \frac{Z_m}{Z_{m+1}}\gamma_m(x_i, y_j) = \frac{Z_m}{Z_{m+1}}\tilde{\mu}_{m+1}(x_i) < \tilde{\mu}_{m+1}(x_i)$, we add $\frac{Z_{m+1}-Z_m}{Z_{m+1}}\tilde{\mu}_{m+1}(x_i)\frac{\tilde{\beta}_{m+1}(y_m)}{\tilde{\beta}_{m+1}(y_m)+\tilde{\beta}_{m+1}(y_{m+1})}$, $\frac{Z_{m+1}-Z_m}{Z_{m+1}}\tilde{\mu}_{m+1}(x_i)\frac{\tilde{\beta}_{m+1}(y_{m+1})}{\tilde{\beta}_{m+1}(y_m)+\tilde{\beta}_{m+1}(y_{m+1})}$ to $\gamma_{m+1}(x_i, y_m)$, $\gamma_{m+1}(x_i, y_{m+1})$, respectively. So we have

$$\sum_j \gamma_{m+1}(x_i, y_j) = \tilde{\mu}_{m+1}(x_i),\ i < m.$$

For $i = m, m+1$, since assumption (5) holds, we have $u + v + w = \tilde{\mu}_{m+1}(x_m) + \tilde{\mu}_{m+1}(x_{m+1})$, $\frac{w}{u+v} = \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)}$, it's obvious that $u + v = \tilde{\mu}_{m+1}(x_m), w = \tilde{\mu}_{m+1}(x_{m+1})$, which is

$$\sum_j \gamma_{m+1}(x_i, y_j) = \tilde{\mu}_{m+1}(x_i),\ i = m, m+1.$$

For $j = m, m+1$, we firstly have

$$\begin{aligned}
\sum_{j=m,m+1} \sum_i \gamma_{m+1}(x_i, y_j) &= \sum_j \sum_i \gamma_{m+1}(x_i, y_j) - \sum_{j \neq m,m+1} \sum_i \gamma_{m+1}(x_i, y_j) \\
&= \sum_i \sum_j \gamma_{m+1}(x_i, y_j) - \sum_{j \neq m,m+1} \tilde{\beta}_{m+1}(y_j) \\
&= \sum_i \tilde{\mu}_{m+1}(x_i) - \sum_{j \neq m,m+1} \tilde{\beta}_{m+1}(y_j) \\
&= 1 - (1 - \tilde{\beta}_{m+1}(y_m) - \tilde{\beta}_{m+1}(y_{m+1})) \\
&= \tilde{\beta}_{m+1}(y_m) + \tilde{\beta}_{m+1}(y_{m+1}).
\end{aligned}$$

By definition of γ_{m+1}, we know $\frac{\gamma_{m+1}(x_i, y_m)}{\gamma_m(x_i, y_m)} = \frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)}$ for $i < m$. By assumption (5), we know $\frac{v+w}{u} = \frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)}$. Combining three equations above together, we have

$$\sum_i \gamma_{m+1}(x_i, y_j) = \tilde{\beta}_{m+1}(y_j),\ j = m, m+1.$$

Now we only need to prove assumption (5) holds. With linear algebra, we solve (5) and have

$$\begin{cases}
u = w \dfrac{1 + \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)}}{\frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)} \left(1 + \frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)}\right)}, \\
v = w \dfrac{\frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)} - \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)}}{\frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)} \left(1 + \frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)}\right)}, \\
w = \dfrac{(\tilde{\mu}_{m+1}(x_m) + \tilde{\mu}_{m+1}(x_{m+1})) \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)} \left(1 + \frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)}\right)}{\left(1 + \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)}\right) \left(1 + \frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)}\right)}.
\end{cases}$$

It's obvious that $u, w \geq 0$. $v \geq 0$ also holds, because

$$\begin{aligned}
\frac{\tilde{\beta}_{m+1}(x_{m+1})}{\tilde{\beta}_{m+1}(x_m)} - \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)} &= \frac{\tilde{\mu}_{m+1}(x_{m+1}) \exp(x_{m+1})}{\tilde{\mu}_{m+1}(x_m) \exp(x_m)} - \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)} \\
&= \frac{\tilde{\mu}_{m+1}(x_{m+1})}{\tilde{\mu}_{m+1}(x_m)} (\exp(x_{m+1} - x_m) - 1) \geq 0.
\end{aligned} \tag{6}$$

So we can find a proper solution of assumption (5).

So γ_{m+1} defined above satisfies UTTC for $(\tilde{\mu}_{m+1}, \tilde{\beta}_{m+1})$.

By induction, for any $0 = x_0 < x_1 < \cdots < x_n < 1$, there exists γ s.t. UTTC (3) holds for $(\tilde{\mu}, \tilde{\beta})$.

Then for any monotonic increasing function, since $\gamma(x_i, y_j) = 0$ when $i > j$, we know $\gamma(x_i, y_j) f(x_i) \leq \gamma(x_i, y_j) f(y_j)$. Hence we have

$$\begin{aligned}
\mathbf{E}_{\tilde{\mu}}[f] = \sum_i \tilde{\mu}(x_i) f(x_i) &= \sum_i \sum_j \gamma(x_i, y_j) f(x_i) \\
&\leq \sum_i \sum_j \gamma(x_i, y_j) f(y_j) \\
&= \sum_j \sum_i \gamma(x_i, y_j) f(y_j) \\
&= \sum_j \tilde{\beta}(y_j) f(y_j) = \mathbf{E}_{\tilde{\beta}}[f].
\end{aligned}$$

$\square$

Lemma 2 (Discretized Upper Triangular Transport Inequality for Co-Monotonic Functions in $\mathbf{R}^1$). *Assume μ is a continuous probability measure supported on $[0,1]$. Let $0 = x_0 < x_1 < \cdots < x_n < 1$ to be any split of $[0,1]$. Let $f, g : \{x_i\}_{i=0,\ldots,n} \to$ **R** to be two co-monotonic functions that satisfy*

$$(f(x_i) - f(x_j)) \cdot (g(x_i) - g(x_j)) \geq 0, \; \forall \, i, j.$$

Define $\tilde{\mu}(x_i) = \mu([x_i, x_{i+1}))$. Define

$$\tilde{\beta}(x_i) = \tilde{\mu}(x_i) \exp(g(x_i))/Z, \; Z = \sum_i \tilde{\mu}(x_i) \exp(g(x_i)).$$

Then we have

$$\boldsymbol{E}_{\tilde{\mu}}[f] \leq \boldsymbol{E}_{\tilde{\beta}}[f].$$

Proof of Lemma 2. If the Upper Triangular Transport Condition (UTTC) holds for $(\tilde{\mu}, \tilde{\beta})$, i.e. there exists a probability measure $\gamma : \{x_i\}_{i=0,\ldots,n} \times \{x_i\}_{i=0,\ldots,n} \to [0,1]$, s.t.

$$\begin{cases} \sum_j \gamma(x_i, y_j) = \tilde{\mu}(x_i), & i = 0, \ldots, n; \\ \sum_i \gamma(x_i, y_j) = \tilde{\beta}(y_j), & j = 0, \ldots, n; \\ \gamma(x_i, y_j) = 0, & g(x_i) > g(y_j), \end{cases}$$

then we finish the proof by

$$\begin{aligned} \mathbf{E}_{\tilde{\mu}}[f] &= \sum_i \tilde{\mu}(x_i) f(x_i) = \sum_i \sum_j \gamma(x_i, y_j) f(x_i) \\ &\leq \sum_i \sum_j \gamma(x_i, y_j) f(y_j) \\ &= \sum_j \sum_i \gamma(x_i, y_j) f(y_j) \\ &= \sum_j \tilde{\beta}(y_j) f(y_j) = \mathbf{E}_{\tilde{\beta}}[f], \end{aligned}$$

where $\gamma(x_i, y_j) f(x_i) \leq \gamma(x_i, y_j) f(y_j)$ is because of $\gamma(x_i, y_j) = 0, \; g(x_i) > g(y_j)$ and $(f(x_i) - f(x_j)) \cdot (g(x_i) - g(x_j)) \geq 0$.

Now we only need to prove UTTC holds for $(\tilde{\mu}, \tilde{\beta})$.

Given $0 = x_0 < x_1 < \cdots < x_n < 1$, we prove the existence of γ by induction. With g to be the transition function in the definition of $\tilde{\beta}$, we mimic the proof of **Lemma** 1 and sort $(x_0, \ldots, x_n)$ in the increasing order of g, which is

$$g(x_{k_0}) \leq g(x_{k_1}) \leq \cdots \leq g(x_{k_n}).$$

Define

$$\tilde{\mu}_m(x_{k_i}) = \begin{cases} \mu([x_{k_i}, \min\{1, x_{k_l} | \, x_{k_l} > x_{k_i}, l \leq m\})), & i \leq m, \; x_{k_i} \neq \min\{x_{k_l} | \, l \leq m\}, \\ \mu([0, \min\{1, x_{k_l} | \, x_{k_l} > x_{k_i}, l \leq m\})), & i \leq m, \; x_{k_i} = \min\{x_{k_l} | \, l \leq m\}, \\ 0, & i > m. \end{cases}$$

Define

$$\tilde{\beta}_m(x_{k_i}) = \tilde{\mu}_m(x_{k_i}) \exp(g(x_{k_i}))/Z_m, \; Z_m = \sum_i \tilde{\mu}_m(x_{k_i}) \exp(g(x_{k_i})).$$

To clarify the proof, we use x_{k_i} to represent the point for $\tilde{\mu}$-axis in coupling and y_{k_j} to represent the point for $\tilde{\beta}$-axis, but they are actually identical, i.e. $x_{k_i} = y_{k_j}$ when $i = j$.

When $m=0$, it's obvious that $(\tilde{\mu}_0, \tilde{\beta}_0)$ satisfies UTTC, as

$$\gamma_0(x_{k_i}, y_{k_j}) = \begin{cases} 1, & i=0, j=0, \\ 0, & else. \end{cases}$$

Assume UTTC holds for m, i.e. there exists γ_m s.t. $(\tilde{\mu}_m, \tilde{\beta}_m)$ satisfies UTTC, we want to prove it also holds for $m+1$.

When $x_{k_{m+1}} > \min\{x_{k_l} | \, l \leq m\}$, let $x_{k^*} = \max\{x_{k_l} | \, x_{k_l} < x_{k_{m+1}}, l \leq m\}$ to be the closest left neighbor of $x_{k_{m+1}}$ in $\{x_{k_l} | \, l \leq m\}$. Then we have $\tilde{\mu}_m(x_{k^*}) = \tilde{\mu}_{m+1}(x_{k^*}) + \tilde{\mu}_{m+1}(x_{k^{m+1}})$.

When $x_{k_{m+1}} < \min\{x_{k_l} | \, l \leq m\}$, let $x_{k^*} = \min\{x_{k_l} | \, l \leq m\}$ to be the leftmost point in $\{x_{k_l} | \, l \leq m\}$. Then we have $\tilde{\mu}_m(x_{k^*}) = \tilde{\mu}_{m+1}(x_{k^*}) + \tilde{\mu}_{m+1}(x_{k^{m+1}})$.

In either case, we always have $\tilde{\mu}_m(x_{k^*}) = \tilde{\mu}_{m+1}(x_{k^*}) + \tilde{\mu}_{m+1}(x_{k_{m+1}})$. By definition of $\tilde{\mu}_m$ and $\tilde{\beta}_m$, we have

$$\begin{cases} \tilde{\mu}_m(x_{k_i}) = \tilde{\mu}_{m+1}(x_{k_i}), & i \leq m, \ k_i \neq k^*, \\ \tilde{\mu}_m(x_{k_i}) = \tilde{\mu}_{m+1}(x_{k_i}) + \tilde{\mu}_{m+1}(x_{k_{m+1}}), & i \leq m, \ k_i = k^*, \\ \tilde{\mu}_m(x_{k_{m+1}}) = \tilde{\mu}_m(x_{k_i}) = \tilde{\mu}_{m+1}(x_{k_i}) = 0, & i > m+1, \end{cases}$$

$$\begin{cases} \tilde{\beta}_m(x_{k_i}) = \tilde{\beta}_{m+1}(x_{k_i}) \cdot \dfrac{Z_{m+1}}{Z_m}, & i \leq m, \ k_i \neq k^*, \\ \tilde{\beta}_m(x_{k_i}) = \Big(\tilde{\beta}_{m+1}(x_{k_i}) + \tilde{\beta}_{m+1}(x_{k_{m+1}}) \exp\big(g(x_{k_i}) - g(x_{k_{m+1}})\big)\Big) \cdot \dfrac{Z_{m+1}}{Z_m}, & i \leq m, \ k_i = k^*, \\ \tilde{\beta}_m(x_{m+1}) = \tilde{\beta}_m(x_i) = \tilde{\beta}_{m+1}(x_i) = 0, & i > m+1. \end{cases}$$

If $g(x_{k^*}) = g(x_{k_{m+1}})$, it's easy to check that $\frac{\tilde{\mu}_{m+1}(x_{k_{m+1}})}{\tilde{\mu}_{m+1}(x_{k^*})} = \frac{\tilde{\beta}_{m+1}(x_{k_{m+1}})}{\tilde{\beta}_{m+1}(x_{k^*})}$, we can simply define the following γ_{m+1} which achieves UTTC for $(\tilde{\mu}_{m+1}, \tilde{\beta}_{m+1})$:

$$\begin{cases} \gamma_{m+1}(x_{k^*}, y_{k_j}) = \gamma_m(x_{k^*}, y_{k_j}) \dfrac{\tilde{\mu}_{m+1}(x_{k^*})}{\tilde{\mu}_{m+1}(x_{k^*}) + \tilde{\mu}_{m+1}(x_{k_{m+1}})}, & j \leq m, \ k_j \neq k^*, \\ \gamma_{m+1}(x_{k_{m+1}}, y_{k_j}) = \gamma_m(x_{k_{m+1}}, y_{k_j}) \dfrac{\tilde{\mu}_{m+1}(x_{k_{m+1}})}{\tilde{\mu}_{m+1}(x_{k^*}) + \tilde{\mu}_{m+1}(x_{k_{m+1}})}, & j \leq m, \ k_j \neq k^*, \\ \gamma_{m+1}(x_{k_i}, y_{k^*}) = \gamma_m(x_{k_i}, y_{k^*}) \dfrac{\tilde{\beta}_{m+1}(y_{k^*})}{\tilde{\beta}_{m+1}(y_{k^*}) + \tilde{\beta}_{m+1}(y_{k_{m+1}})}, & i \leq m, \ k_i \neq k^*, \\ \gamma_{m+1}(x_{k_i}, y_{k_{m+1}}) = \gamma_m(x_{k_i}, y_{k_{m+1}}) \dfrac{\tilde{\beta}_{m+1}(y_{k_{m+1}})}{\tilde{\beta}_{m+1}(y_{k^*}) + \tilde{\mu}_{m+1}(x_{k_{m+1}})}, & i \leq m, \ k_i \neq k^*, \\ \gamma_{m+1}(x_{k^*}, y_{k^*}) = \gamma_m(x_{k^*}, y_{k^*}) \dfrac{\tilde{\mu}_{m+1}(x_{k^*})}{\tilde{\mu}_{m+1}(x_{k^*}) + \tilde{\mu}_{m+1}(x_{k_{m+1}})}, & \\ \gamma_{m+1}(x_{k_{m+1}}, y_{k_{m+1}}) = \gamma_m(x_{k_{m+1}}, y_{k_{m+1}}) \dfrac{\tilde{\mu}_{m+1}(x_{k_{m+1}})}{\tilde{\mu}_{m+1}(x_{k^*}) + \tilde{\mu}_{m+1}(x_{k_{m+1}})}, & \\ \gamma_{m+1}(x_{k_i}, y_{k_j}) = 0, & others. \end{cases}$$

If $g(x_{k^*}) < g(x_{k_{m+1}})$, recalling the proof of **Lemma** 1, it's crucial to prove inequalities (4) and (6). Inequality (4) guarantees that $\frac{Z_m}{Z_{m+1}} < 1$, so we can shrinkage γ_m entrywise by $\frac{Z_m}{Z_{m+1}}$ and add some proper measure at proper points. Inequality (6) guarantees that (x_m, y_m) can be decomposed to (x_m, y_m), (x_m, y_{m+1}), (x_{m+1}, y_{m+1}). Following the idea, we check that

$$Z_{m+1} - Z_m = \tilde{\mu}_{m+1}(x_{k_{m+1}}) \big(\exp(g(x_{k_{m+1}}) - g(x_{k^*}))\big) > 0,$$

$$\begin{aligned} \frac{\tilde{\beta}_{m+1}(x_{k_{m+1}})}{\tilde{\beta}_{m+1}(x_{k^*})} - \frac{\tilde{\mu}_{m+1}(x_{k_{m+1}})}{\tilde{\mu}_{m+1}(x_{k^*})} &= \frac{\tilde{\mu}_{m+1}(x_{k_{m+1}}) \exp(g(x_{k_{m+1}}))}{\tilde{\mu}_{m+1}(x_{k^*}) \exp(g(x_{k^*}))} - \frac{\tilde{\mu}_{m+1}(x_{k_{m+1}})}{\tilde{\mu}_{m+1}(x_{k^*})} \\ &= \frac{\tilde{\mu}_{m+1}(x_{k_{m+1}})}{\tilde{\mu}_{m+1}(x_{k^*})} \big(\exp(g(x_{k_{m+1}}) - g(x_{k^*})) - 1\big) > 0. \end{aligned}$$

Replacing x_m, x_{m+1} in the proof of **Lemma** 1 by $x_{k^*}, x_{k_{m+1}}$, we can construct γ_{m+1} all the same way as in the proof of **Lemma** 1.

By induction, we prove UTTC for $(\tilde{\mu}, \tilde{\beta})$. The proof is done. $\square$

Theorem 3 (Upper Triangular Transport Inequality for Co-Monotonic Functions in $\mathbf{R}^1$). *Assume μ is a continuous probability measure supported on $[0,1]$. Let $f, g : [0,1] \to \boldsymbol{R}$ to be two co-monotonic functions that satisfy*

$$(f(x) - f(y)) \cdot (g(x) - g(y)) \geq 0, \ \forall\, x, y \in [0,1].$$

f is continuous. Define

$$\beta(x) = \mu(x)\exp(g(x))/Z, \ Z = \int_{[0,1]} \mu(x)\exp(g(x)).$$

Then we have

$$\boldsymbol{E}_{\mu}[f] \leq \boldsymbol{E}_{\beta}[f].$$

Proof of Theorem 3. For $\forall \epsilon > 0$, since f is continuous, f is uniformly continuous, so there exists $\delta > 0$ s.t. $|f(x) - f(y)| < \epsilon, \forall x, y \in [0,1]$. We can split $[0,1]$ by $0 < x_0 < x_1 < \cdots < x_n < 1$ s.t. $x_{i+1} - x_i < \delta$. Define $\tilde{\mu}$ and $\tilde{\beta}$ as in **Lemma** 2. Since $x_{i+1} - x_i < \delta$, by uniform continuity and the definition of the expectation, we have

$$|\mathbf{E}_{\mu}[f] - \mathbf{E}_{\tilde{\mu}}[f]| < \epsilon, \ |\mathbf{E}_{\beta}[f] - \mathbf{E}_{\tilde{\beta}}[f]| < \epsilon,$$

By **Lemma** 2, we have

$$\mathbf{E}_{\tilde{\mu}}[f] \leq \mathbf{E}_{\tilde{\beta}}[f].$$

So we have

$$\mathbf{E}_{\mu}[f] < \mathbf{E}_{\tilde{\mu}}[f] + \epsilon \leq \mathbf{E}_{\tilde{\beta}}[f] + \epsilon < \mathbf{E}_{\beta}[f] + 2\epsilon.$$

Since ϵ is arbitrary, we prove $\mathbf{E}_{\mu}[f] \leq \mathbf{E}_{\beta}[f]$.

$\square$

Lemma 3 (Discretized Upper Triangular Transport Inequality for Co-Monotonic Functions in $\mathbf{R}^p$). *Assume μ is a continuous probability measure supported on $[0,1]^p$. Let $0 = x_0^d < x_1^d < \cdots < x_n^d < 1$ to be any split of $[0,1]$, $d = 1, \ldots, p$. Denote $\boldsymbol{x_i} \stackrel{def}{=} (x_{i_1}^1, \ldots, x_{i_p}^p)$. Define $\tilde{\mu}(\boldsymbol{x_i}) = \mu(\prod_{d=1,\ldots,p}[x_{i_d}^d, x_{i_d+1}^d))$. Let $f, g : \{\boldsymbol{x_i}\}_{i \in \{0,\ldots,n\}^p} \to \boldsymbol{R}$ to be two co-monotonic functions that satisfy*

$$(f(\boldsymbol{x_i}) - f(\boldsymbol{x_j})) \cdot (g(\boldsymbol{x_i}) - g(\boldsymbol{x_j})) \geq 0, \ \forall\, \boldsymbol{i}, \boldsymbol{j}.$$

Define

$$\tilde{\beta}(\boldsymbol{x_i}) = \tilde{\mu}(\boldsymbol{x_i})\exp(g(\boldsymbol{x_i}))/Z, \ Z = \sum_{\boldsymbol{i}} \tilde{\mu}(\boldsymbol{x_i})\exp(g(\boldsymbol{x_i})).$$

Then there exists a probability measure $\gamma : \{\boldsymbol{x_i}\}_{i \in \{0,\ldots,n\}^p} \times \{\boldsymbol{x_j}\}_{j \in \{0,\ldots,n\}^p} \to [0,1]$, s.t.

$$\sum_{\boldsymbol{j}} \gamma(\boldsymbol{x_i}, \boldsymbol{y_j}) = \tilde{\mu}(\boldsymbol{x_i}), \ \forall\, \boldsymbol{i};$$
$$\sum_{\boldsymbol{i}} \gamma(\boldsymbol{x_i}, \boldsymbol{y_j}) = \tilde{\beta}(\boldsymbol{y_j}), \ \forall\, \boldsymbol{j};$$
$$\gamma(\boldsymbol{x_i}, \boldsymbol{y_j}) = 0, \ g(\boldsymbol{x_i}) > g(\boldsymbol{y_j}).$$

Then we have

$$\boldsymbol{E}_{\tilde{\mu}}[f] \leq \boldsymbol{E}_{\tilde{\beta}}[f].$$

Proof of Lemma 3. The proof is almost identical to the proof of **Lemma** 2, except for the definition of $(\tilde{\mu}_m, \tilde{\beta}_m)$ in $\mathbf{R}^p$.

Given $\{\mathbf{x_i}\}_{\mathbf{i}\in\{0,\dots,n\}^p}$, we sort $\mathbf{x_i}$ in the increasing order of g, which is

$$g(\mathbf{x}_{\mathbf{k}_0}) \le g(\mathbf{x}_{\mathbf{k}_1}) \le \dots \le g(\mathbf{x}_{\mathbf{k}_{(n+1)^p-1}}),$$

where $\{\mathbf{k}_i\}_{i\in\{0,\dots,(n+1)^p-1\}}$ is a permutation of $\{\mathbf{i}\}_{\mathbf{i}\in\{0,\dots,n\}^p}$.

For $\mathbf{i},\mathbf{j} \in \{0,\dots,n\}^p$, we define the partial order $\mathbf{i} < \mathbf{j}$ on $\{0,\dots,n\}^p$, if

$$\exists 0 \le d_0 \le n,\ s.t.\ \mathbf{i}_d \le \mathbf{j}_d,\ \forall d < d_0\ and\ \mathbf{i}_{d_0} < \mathbf{j}_{d_0}.$$

It's obvious that

$$\begin{cases} \forall \mathbf{i} \in \{0,\dots,n\}^p,\ \mathbf{i} \not< \mathbf{i}, \\ \forall \mathbf{i},\mathbf{j} \in \{0,\dots,n\}^p,\ \mathbf{i} < \mathbf{j} \Rightarrow \mathbf{j} \not< \mathbf{i}, \\ \forall \mathbf{i},\mathbf{j},\mathbf{k} \in \{0,\dots,n\}^p,\ \mathbf{i} < \mathbf{j},\ \mathbf{j} < \mathbf{k} \Rightarrow \mathbf{i} < \mathbf{k}. \end{cases}$$

We define $\mathbf{i} = \mathbf{j}$ if $\mathbf{i}_d = \mathbf{j}_d$, $\forall\, 0 \le d \le n$. So we define the partial order relation, and we can further define the $\min$ function and the $\max$ function on $\{0,\dots,n\}^p$.

Now using this partial order relation, we define

$$\tilde{\mu}_m(\mathbf{x}_{\mathbf{k}_i}) = \begin{cases} \sum\limits_{\mathbf{k}\ge\mathbf{k}_i,\mathbf{k}<\min\{\mathbf{k}_l|\mathbf{k}_l>\mathbf{k}_i,l\le m\}} \tilde{\mu}(\mathbf{x_k}), & i \le m,\ \mathbf{k}_i \ne \min\{\mathbf{k}_l|\ l \le m\}, \\ \sum\limits_{\mathbf{k}<\min\{\mathbf{k}_l|\mathbf{k}_l>\mathbf{k}_i,l\le m\}} \tilde{\mu}(\mathbf{x_k}), & i \le m,\ \mathbf{k}_i = \min\{\mathbf{k}_l|\ l \le m\}, \\ 0, & i > m. \end{cases}$$

With this definition of $\tilde{\mu}_m$, other parts are identical to the proof of **Lemma** 2. The proof is done.

$\square$

Theorem 4 (Upper Triangular Transport Inequality for Co-Monotonic Functions in $\mathbf{R}^p$). *Assume μ is a continuous probability measure supported on $[0,1]^p$. Denote $\boldsymbol{x} \overset{def}{=} (x^1,\dots,x^p)$. Let $f,g : [0,1]^p \to \boldsymbol{R}$ to be two co-monotonic functions that satisfy*

$$(f(\boldsymbol{x}) - f(\boldsymbol{y})) \cdot (g(\boldsymbol{x}) - g(\boldsymbol{y})) \ge 0,\ \forall\, \boldsymbol{x},\boldsymbol{y} \in [0,1]^p.$$

f is continuous. Define

$$\beta(\boldsymbol{x}) = \mu(\boldsymbol{x}) \exp(g(\boldsymbol{x}))/Z,\ Z = \int_{[0,1]^p} \mu(\boldsymbol{x}) \exp(g(\boldsymbol{x})).$$

Let $f,g : [0,1]^p \to \boldsymbol{R}$ to be two co-monotonic functions that satisfy

$$(f(\boldsymbol{x}) - f(\boldsymbol{y})) \cdot (g(\boldsymbol{x}) - g(\boldsymbol{y})) \ge 0,\ \forall\, \boldsymbol{x},\boldsymbol{y} \in [0,1]^p.$$

Then we have

$$\boldsymbol{E}_\mu[f] \le \boldsymbol{E}_\beta[f].$$

Proof of Theorem 4. For $\forall \epsilon > 0$, since f is continuous, f is uniformly continuous, so there exists $\delta > 0$ s.t. $|f(\mathbf{x}) - f(\mathbf{y})| < \epsilon, \forall \mathbf{x},\mathbf{y} \in [0,1]^p$. We can split $[0,1]$ by $0 < x_0 < x_1 < \dots < x_n < 1$ s.t. $x_{i+1} - x_i < \delta/\sqrt{p}$. Define $x_i^d = x_i,\ \forall 0 \le d \le p$. Define $\tilde{\mu}$ and $\tilde{\beta}$ as in **Lemma** 3. Since $x_{i+1} - x_i < \delta/\sqrt{p}$, $|(x_{i+1}^0,\dots,x_{i+1}^p) - (x_i^0,\dots,x_i^p)| < \delta$, by uniform continuity and the definition of the expectation, we have

$$|\mathbf{E}_\mu[f] - \mathbf{E}_{\tilde{\mu}}[f]| < \epsilon,\ |\mathbf{E}_\beta[f] - \mathbf{E}_{\tilde{\beta}}[f]| < \epsilon,$$

By **Lemma** 3, we have

$$\mathbf{E}_{\tilde{\mu}}[f] \le \mathbf{E}_{\tilde{\beta}}[f].$$

So we have

$$\mathbf{E}_{\mu}[f] < \mathbf{E}_{\tilde{\mu}}[f] + \epsilon \leq \mathbf{E}_{\tilde{\beta}}[f] + \epsilon < \mathbf{E}_{\beta}[f] + 2\epsilon.$$

Since ϵ is arbitrary, we prove $\mathbf{E}_{\mu}[f] \leq \mathbf{E}_{\beta}[f]$.

$\square$

Lemma 4 (Performance Difference Lemma)**.** *For any policies π, π' and any state s_0, we have*

$$V^{\pi}(s_0) - V^{\pi'}(s_0) = \frac{1}{1-\gamma} \boldsymbol{E}_{s \sim d_{s_0}^{\pi}} \boldsymbol{E}_{a \sim \pi(\cdot|s)} \left[A^{\pi'}(s,a) \right].$$

Proof. See (Kakade & Langford, 2002). $\square$

D. Algorithm Pseudocode

D.1. GDI-I^3

In this section, we provide the implementation pseudocode of GDI-I^3, which is shown in **Algorithm** 2.

$$\begin{cases} A = A_\theta\left(s_t\right), & V = V_\theta\left(s_t\right), \\ \bar{A} = A - E_\pi[A], & Q = \bar{A} + V. \end{cases} \tag{7}$$

$$\lambda = (\tau_1, \tau_2, \epsilon),\ \pi_{\theta_\lambda} = \epsilon \cdot \underbrace{\operatorname{Softmax}\left(\frac{A}{\tau_1}\right)}_{Exploration} + (1-\epsilon) \cdot \underbrace{\operatorname{Softmax}\left(\frac{A}{\tau_2}\right)}_{Exploitation} \tag{8}$$

Algorithm 2 GDI-I^3 Algorithm.

Initialize Parameter Server (PS) and Data Collector (DC).

// LEARNER
Initialize d_{push}.
Initialize θ as Eq. (7) and (8).
Initialize $count = 0$.
while $True$ **do**
  Load data from DC.
  Estimate qs and vs by proper off-policy algorithms.
    (For instance, ReTrace (B.1) for qs and V-Trace (B.2) for vs.)
  Update θ via policy gradient and policy evaluation.
  if $count$ mod d_{push} = 0 **then**
    Push θ to PS.
  end if
  $count \leftarrow count + 1$.
end while

// ACTOR
Initialize d_{pull}, M.
Initialize θ as Eq. (7) and (8).
Initialize $\{\mathcal{B}_m\}_{m=1,...,M}$ and sample λ as in **Algorithm** 4.
Initialize $count = 0$, $G = 0$.
while $True$ **do**
  Calculate $\pi_{\theta_\lambda}(\cdot|s)$.
  Sample $a \sim \pi_{\theta_\lambda}(\cdot|s)$.
  $s, r, done \sim p(\cdot|s, a)$.
  $G \leftarrow G + r$.
  if $done$ **then**
    Update $\{\mathcal{B}_m\}_{m=1,...,M}$ with (λ, G) as in **Algorithm** 4.
    Send data to DC and reset the environment.
    $G \leftarrow 0$.
    Sample λ as in **Algorithm** 4
  end if
  if $count \mod d_{pull}$ = 0 **then**
    Pull θ from PS and update θ.
  end if
  $count \leftarrow count + 1$.
end while

D.2. GDI-H^3

In this section, we provide the implementation pseudocode of GDI-H^3, which is shown in **Algorithm** 3.

$$
\begin{cases} A_{\theta_1} = A_{\theta_1}\left(s_t\right), & V_{\theta_1} = V_{\theta_1}\left(s_t\right), \\ \bar{A}_{\theta_1} = A_{\theta_1} - E_\pi[A_{\theta_1}], & Q_{\theta_1} = \bar{A}_{\theta_1} + V_{\theta_1}. \end{cases}
\begin{cases} A_{\theta_2} = A_{\theta_2}\left(s_t\right), & V_{\theta_2} = V_{\theta_2}\left(s_t\right), \\ \bar{A}_{\theta_2} = A_{\theta_2} - E_\pi[A_{\theta_2}], & Q_{\theta_2} = \bar{A}_{\theta_2} + V_{\theta_2}. \end{cases} \tag{9}
$$

$$
\lambda = (\tau_1, \tau_2, \epsilon),\ \pi_{\theta_\lambda} = \epsilon \cdot \mathrm{Softmax}\left(\frac{A_{\theta_1}}{\tau_1}\right) + (1-\epsilon) \cdot \mathrm{Softmax}\left(\frac{A_{\theta_2}}{\tau_2}\right) \tag{10}
$$

Algorithm 3 GDI-H^3 Algorithm.

Initialize Parameter Server (PS) and Data Collector (DC).

// LEARNER
Initialize d_{push}.
Initialize θ as Eq. (9) and (10).
Initialize $count = 0$.
while $True$ **do**
  Load data from DC.
  Estimate qs_1, qs_2 and vs_1, vs_2 by proper off-policy algorithms.
    (For instance, ReTrace (B.1) for $qs1, qs_2$ and V-Trace (B.2) for vs_1, vs_2.)
  Update θ_1, θ_2 via policy gradient and policy evaluation, respectively.
  if $count$ mod d_{push} = 0 **then**
    Push θ_1, θ_2 to PS.
  end if
  $count \leftarrow count + 1$.
end while

// ACTOR
Initialize d_{pull}, M.
Initialize θ_1, θ_2 as Eq. (9) and (10).
Initialize $\{\mathcal{B}_m\}_{m=1,...,M}$ and sample λ as in **Algorithm** 4.
Initialize $count = 0$, $G = 0$.
while $True$ **do**
  Calculate $\pi_{\theta_\lambda}(\cdot|s)$.
  Sample $a \sim \pi_{\theta_\lambda}(\cdot|s)$.
  $s, r, done \sim p(\cdot|s, a)$.
  $G \leftarrow G + r$.
  if $done$ **then**
    Update $\{\mathcal{B}_m\}_{m=1,...,M}$ with (λ, G) as in **Algorithm** 4.
    Send data to DC and reset the environment.
    $G \leftarrow 0$.
    Sample λ as in **Algorithm** 4
  end if
  if $count \mod d_{pull}$ = 0 **then**
    Pull θ from PS and update θ.
  end if
  $count \leftarrow count + 1$.
end while

E. Adaptive Controller Formalism

In practice, we use a Bandits Controller (BC) to control the behavior sampling distribution adaptively, which has been widely used in prior works (Badia et al., 2020a; Xiao et al., 2021b). More details on Bandits can see (Sutton & Barto, 2018). The whole algorithm is shown in **Algorithm** 4. As the behavior policy can be parameterized and thereby sampling behaviors from the policy space is equivalent to sampling indexes x from the index set.

Let's firstly define a bandit as $B = Bandit(mode, l, r, lr, d, acc, ta, to, \mathbf{w}, \mathbf{N})$.

- $mode$ is the mode of sampling, with two choices, $argmax$ and $random$, wherein $argmax$ greedily chooses the behaviors with top estimated value from the policy space, and $random$ samples behaviors according to a distribution calculated by $Softmax(V)$.
- l is the left boundary of the index set, and each x is clipped to $x = \max\{x, l\}$.
- r is the right boundary of the index set, and each x is clipped to $x = \min\{x, r\}$.
- acc is the accuracy of space to be optimized, where each x is located in the $\lfloor(\min\{\max\{x, l\}, r\} - l)/acc\rfloor$th block.
- tile coding is a representation method of continuous space (Sutton & Barto, 2018), and each kind of tile coding can be uniquely determined by l, r, to and ta, wherein to represents the tile offset and ta represents the accuracy of the tile coding.
- to is the offset of each tile coding, which represents the relative offset of the basic coordinate system (normally we select the space to be optimized as basic coordinate system).
- ta is the accuracy of each tile coding, where each x is located in the $\lfloor(\min\{\max\{x - to, l\}, r\} - l)/ta\rfloor$th tile.
- M_{btt} represents block-to-tile, which is a mapping from the block of the original space to the tile coding space.
- M_{ttb} represents tile-to-block, which is a mapping from the tile coding space to the block of the original space.
- $\mathbf{w}$ is a vector in $\mathbf{R}^{\lfloor(r-l)/ta\rfloor}$, which represents the weight of each tile.
- $\mathbf{N}$ is a vector in $\mathbf{R}^{\lfloor(r-l)/ta\rfloor}$, which counts the number of sampling of each tile.
- lr is the learning rate.
- d is an integer, which represents how many candidates is provided by each bandit when sampling.

During the evaluation process, we evaluate the value of the ith tile by

$$V_i = \frac{\sum_k^{M_{btt}(block_i)} \mathbf{w}_k}{len(M_{btt}(block_i))} \tag{11}$$

During the training process, for each sample (x, g), where g is the target value. Since x locates in the jth tile of kth tile_coding, we update B by

$$\begin{cases} j = \lfloor(\min\{\max\{x - to_k, l\}, r\} - l)/ta_k\rfloor, \\ \mathbf{w}_j \leftarrow \mathbf{w}_j + lr * (g - V_i) \\ \mathbf{N}_j \leftarrow \mathbf{N}_j + 1 \end{cases} \tag{12}$$

During the sampling process, we firstly evaluate $\mathcal{B}$ by (11) and get $(V_1, ..., V_{\lfloor(r-l)/acc\rfloor})$. We calculate the score of ith tile by

$$score_i = \frac{V_i - \mu(\{V_j\}_{j=1,...,\lfloor(r-l)/acc\rfloor})}{\sigma(\{V_j\}_{j=1,...,\lfloor(r-l)/acc\rfloor})} + c \cdot \sqrt{\frac{\log(1 + \sum_j \mathbf{N}_j)}{1 + \mathbf{N}_i}}. \tag{13}$$

For different $mode$s, we sample the candidates by the following mechanism,

- if $mode$ = $argmax$, find blocks with top-d $score$s, then sample d candidates from these blocks, one uniformly from a block;
- if $mode$ = $random$, sample d blocks with $score$s as the logits without replacement, then sample d candidates from these blocks, one uniformly from a block;

In practice, we define a set of bandits $\mathcal{B}_m = \{B_m\}_{m=1,...,M}$. At each step, we sample d candidates $\{c_{m,i}\}_{i=1,...,d}$ from each B_m, so we have a set of $m \times d$ candidates $\{c_{m,i}\}_{m=1,...,M;i=1,...,d}$. Then we sample uniformly from these $m \times d$ candidates to get x. At last, we transform the selected x to $\alpha = \{\tau_1, \tau_2, \epsilon\}$ by $\tau_{1,2} = \frac{1}{\exp(x_{1,2})-1}$ and $\epsilon = x_3$ When we receive (α, g), we transform α to x by $x_{1,2} = \log(1 + 1/\tau_{1,2})$, and $x_3 = \epsilon$. Then we update each B_m by (12).

Algorithm 4 Bandits Controller

```
for m = 1, ..., M do
  Sample mode ~ {argmax, random} and other initialization parameters
  Initialize B_m = Bandit(mode, l, r, lr, d, acc, to, ta, w, N)
  Ensemble B_m to constitute 𝓑_m
end for
while True do
  for m = 1, ..., M do
    Evaluate 𝓑_m by (11).
    Sample candidates c_{m,1}, ..., c_{m,d} from 𝓑_m via (13) following its mode.
  end for
  Sample x from {c_{m,i}}_{m=1,...,M;i=1,...,d}.
  Execute x and receive the return G.
  for m = 1, ..., M do
    Update 𝓑_m with (x, G) by (12).
  end for
end while
```

F. Experiment Details

The overall training architecture is on the top of the Learner-Actor framework (Espeholt et al., 2018), which supports large-scale training. Additionally, the recurrent encoder with LSTM (Schmidhuber, 1997) is used to handle the partially observable MDP problem (Bellemare et al., 2013). The burn-in technique is adopted to deal with the representational drift (Kapturowski et al., 2018), and we train each sample twice. A complete description of the hyperparameters can see App. G. We employ additional environments to evaluate the scores during training, and the undiscounted episode returns averaged over 32 environments with different seeds have been recorded. Details on relevant evaluation criteria can see App. H.

We evaluated all agents on 57 Atari 2600 games from the arcade learning environment (Bellemare et al., 2013, ALE) by recording the average score of the population of agents during training. We have demonstrated our evaluation metrics for ALE in App. H, and we will describe more details in the following. Besides, all the experiment is accomplished using a single CPU with 92 cores and a single Tesla-V100-SXM2-32GB GPU.

Noting that episodes will be truncated at 100K frames (or 30 minutes of simulated play) as other baseline algorithms (Hessel et al., 2017; Badia et al., 2020a; Schmitt et al., 2020; Badia et al., 2020b; Kapturowski et al., 2018) and thereby we calculate the mean playtime over 57 games which is called Playtime. In addition to comparing the mean and median human normalized scores (HNS), we also report the performance based on human world records among these algorithms and the related learning efficiency to further highlight the significance of our algorithm. Inspired by (Toromanoff et al., 2019), human world records normalized score (HWRNS) and SABER are better descriptors for evaluating algorithms on human top level on Atari games, which simultaneously give rise to more challenges and lead the related research into a new journey to train the superhuman agent instead of just paying attention to the human average level.

G. Hyperparameters

In this section, we firstly detail the hyperparameters we use to pre-process the environment frames received from the Arcade Learning Environment. The hyperparameters that we used in all experiments are almost the same as Agent57 (Badia et al., 2020a), NGU (Badia et al., 2020b), MuZero (Schrittwieser et al., 2020) and R2D2 (Kapturowski et al., 2018). In Tab. 4, we detail these pre-processing hyperparameters. Then we will detail the hyperparameters we used for Atari experiments, which is demonstrated in Tab. 5.

Table 4. Atari pre-processing hyperparameters.

Hyperparameter	**Value**
Random modes and difficulties	No
Sticky action probability	0.0
Life information	Not allowed
Image Size	(84, 84)
Num. Action Repeats	4
Num. Frame Stacks	4
Action Space	Full
Max episode length	100000
Random noops range	30
Grayscaled/RGB	Grayscaled

Table 5. Hyperparameters for Atari experiments.

Parameter	**Value**
Num. Frames	200M (2E+8)
Replay	2
Num. Environments	160
GDI-I^3 Reward Shape	$\log(abs(r)+1.0)\cdot(2\cdot 1_{\{r\geq 0\}}-1_{\{r<0\}})$
GDI-H^3 Reward Shape 1	$\log(abs(r)+1.0)\cdot(2\cdot 1_{\{r\geq 0\}}-1_{\{r<0\}})$
GDI-H^3 Reward Shape 2	$sign(r)\cdot((abs(r)+1.0)^{0.25}-1.0)+0.001\cdot r$
Reward Clip	No
Intrinsic Reward	No
Entropy Regularization	No
Burn-in	40
Seq-length	80
Burn-in Stored Recurrent State	Yes
Bootstrap	Yes
Batch size	64
Discount (γ)	0.997
V-loss Scaling (ξ)	1.0
Q-loss Scaling (α)	10.0
π-loss Scaling (β)	10.0
Importance Sampling Clip $\bar{c}$	1.05
Importance Sampling Clip $\bar{\rho}$	1.05
Backbone	IMPALA,deep
LSTM Units	256
Optimizer	Adam Weight Decay
Weight Decay Rate	0.01
Weight Decay Schedule	Anneal linearly to 0
Learning Rate	5e-4
Warmup Steps	4000
Learning Rate Schedule	Anneal linearly to 0
AdamW β_1	0.9
AdamW β_2	0.98
AdamW ϵ	1e-6
AdamW Clip Norm	50.0
Auxiliary Forward Dynamic Task	Yes
Auxiliary Inverse Dynamic Task	Yes
Learner Push Model Every N Steps	25
Actor Pull Model Every N Steps	64
Num. Bandits	7
Bandit Learning Rate	Uniform([0.05, 0.1, 0.2])
Bandit Tiling Width	Uniform([2, 3, 4])
Num. Bandit Candidates	3
Offset of Tile coding	Uniform([0, 60])
Accuracy of Tile coding	Uniform([2, 3, 4])
Accuracy of Search Range for [$1/\tau_1$,$1/\tau_2$,ϵ]	[1.0, 1.0, 0.1]
Fixed Selection for [$1/\tau_1$,$1/\tau_2$,ϵ]	[1.0,0.0,1.0]
Bandit Search Range for $1/\tau_1$	[0.0, 50.0]
Bandit Search Range for $1/\tau_2$	[0.0, 50.0]
Bandit Search Range for ϵ	[0.0, 1.0]

H. Evaluation Metrics for ALE

In this section, we will mainly introduce the evaluation metrics in ALE, including those that have been commonly used by previous works like the raw score and the normalized score over all the Atari games based on human average score baseline, and some novel evaluation criteria for the superhuman Atari benchmark such as the normalized score based on human world records, learning efficiency, and human world record breakthrough. For the summary of benchmark results on these evaluation metrics can see App. J. For more details on these evaluation metrics, we refer to see (Fan, 2021).

H.1. Raw Score

Raw score refers to using tables (e.g., Table of Scores) or figures (e.g., Training Curve) to show the total scores of RL algorithms on all Atari games, which can be calculated by the sum of the undiscounted reward of the **g**th game of Atari using algorithm **i** as follows:

$$G_{g,i} = \mathbf{E}_{s_t \sim d_{\rho_0}^{\pi}} \mathbf{E}_{\pi} \left[\sum_{k=0}^{\infty} r_{t+k} | s_t \right], g \in [1, 57] \tag{14}$$

As Bellemare et al. (2013) firstly put it, raw score over the whole 57 Atari games can reflect the performance and generality of RL agents to a certain extent. However, this evaluation metric has many limitations:

1. It is difficult to compare the performance of the two algorithms directly.
2. Its value is easily affected by the score scale. For example, the score scale of Pong is [-21,21], but that of Chopper Command is [0,999900], so the Chopper Command will dominate the mean score of those games.

In recent RL advances, this metric is used to avoid any issues that aggregated metrics may have (Badia et al., 2020a). Furthermore, this paper used these metrics to prove whether the RL agents have surpassed the human world records, which will be introduced in detail later.

H.2. Normalized Scores

To handle the drawbacks of the raw score, some methods (Bellemare et al., 2013; Mnih et al., 2015) proposed the normalized score. The normalized score of the **g**th game of Atari using algorithm **i** can be calculated as follows:

$$Z_{g,i} = \frac{G_{g,i} - G_{g,base}}{G_{g,reference} - G_{g,base}} \tag{15}$$

As Bellemare et al. (2013) put it, we can compare games with different scoring scales by normalizing scores, which makes the numerical values become comparable. In practice, we can make $G_{g,base} = r_{g,min}$ and $G_{g,reference} = r_{g,max}$, where $[r_{g,min}, r_{g,max}]$ is the score scale of the **g**th game. Then Equ. (15) becomes $Z_{g,i} = \frac{G_{g,i} - r_{g,min}}{r_{i,max} - r_{g,min}}$, which is a **Min-Max Scaling** and thereby $Z_{g,i} \in [0, 1]$ become comparable across the 57 games. It seems this metric can be served to compare the performance between two different algorithms. However, the Min-Max normalized score fail to intuitively reflect the gap between the algorithm and the average level of humans. Thus, we need a human baseline normalized score.

H.2.1. Human Average Score Baseline

As we mentioned above, recent reinforcement learning advances (Badia et al., 2020a;b; Kapturowski et al., 2018; Ecoffet et al., 2019; Schrittwieser et al., 2020; Hessel et al., 2021; 2017) are seeking agents that can achieve superhuman performance. Thus, we need a metric to intuitively reflect the level of the algorithms compared to human performance. Since being proposed by (Bellemare et al., 2013), the Human Normalized Score (HNS) is widely used in the RL research(Machado et al., 2018). HNS can be calculated as follows:

$$\text{HNS}_{g,i} = \frac{G_{g,i} - G_{g,\text{random}}}{G_{g,\text{human average}} - G_{g,\text{random}}} \tag{16}$$

wherein g denotes the gth game of Atari, i represents the algorithm i, $G_{g,\text{human average}}$ represents the human average score baseline (Toromanoff et al., 2019), and $G_{g,\text{random}}$ represents the performance of a random policy. Adopting HNS as an evaluation metric has the following advantages:

1. **Intuitive comparison with human performance.** $\text{HNS}_{g,i} \geq 100\%$ means algorithm i have surpassed the human average performance in game g. Therefore, we can directly use HNS to reflect which games the RL agents have surpassed the average human performance.

2. **Performance across algorithms become comparable.** Like Max-Min Scaling, the human normalized score can also make two different algorithms comparable. The value of $\text{HNS}_{g,i}$ represents the degree to which algorithm i surpasses the average level of humans in game g.

Mean HNS represents the mean performance of the algorithms across the 57 Atari games based on the human average score. However, it is susceptible to interference from individual high-scoring games like the hard-exploration problems in Atari (Ecoffet et al., 2019). While taking it as the only evaluation metric, Go-Explore(Ecoffet et al., 2019) has achieved SOTA compared to Agent57(Badia et al., 2020a), NGU(Badia et al., 2020b), R2D2(Kapturowski et al., 2018). However, Go-Explore fails to handle many other games in Atari like Demon Attack, Breakout, Boxing, Phoenix. Additionally, Go-Explore fails to balance the trade-off between exploration and exploitation, which makes it suffer from the low sample efficiency problem, which will be discussed later.

Median HNS represents the median performance of the algorithms across the 57 Atari games based on the human average score. Some methods (Schrittwieser et al., 2020; Hessel et al., 2021) have adopted it as a more reasonable metric for comparing performance between different algorithms. The median HNS has overcome the interference from individual high-scoring games. However, As far as we can see, there are at least two problems while only referring to it as the evaluation metrics. First of all, the median HNS only represents the mediocre performance of an algorithm. How about the top performance? One algorithm (Hessel et al., 2021) can easily achieve high median HNS, but at the same time obtain a poor mean HNS by adjusting the hyperparameters of algorithms for games near the median score. It shows that these metrics can show the generality of the algorithms but fail to reflect the algorithm's potential. Moreover, adopting these metrics will urge us to pursue rather mediocre methods.

In practice, we often use **mean HNS** or **median HNS** to show the final performance or generality of an algorithm. Dispute upon whether the mean value or the median value is more representative to show the generality and performance of the algorithms lasts for several years (Mnih et al., 2015; Hessel et al., 2017; Hafner et al., 2020; Hessel et al., 2021; Bellemare et al., 2013; Machado et al., 2018). To avoid any issues that aggregated metrics may have, **we advocate calculating both of them in the final results** because they serve different purposes, and we could not evaluate any algorithm via a single one of them.

H.2.2. Capped Normalized Score

Capped Normalized Score is also widely used in many reinforcement learning advances (Toromanoff et al., 2019; Badia et al., 2020a). Among them, Agent57 (Badia et al., 2020a) adopts the capped human normalized score (CHNS) as a better descriptor for evaluating general performance, which can be calculated as $\text{CHNS} = \max\{\min\{\text{HNS}, 1\}, 0\}$. Agent57 claimed CHNS emphasizes the games that are below the average human performance benchmark and used CHNS to judge whether an algorithm has surpassed the human performance via $\text{CHNS} \geq 100\%$. The mean/median CHNS represents the mean/median completeness of surpassing human performance. However, there are several problems while adopting these metrics:

1. CHNS fails to reflect the real performance in specific games. For example, $\text{CHNS} \geq 100\%$ represents the algorithms surpassed the human performance but failed to reveal how good the algorithm is in this game. From the view of CHNS, Agent57 (Badia et al., 2020a) has achieved SOTA performance across 57 Atari games, but while referring to the mean HNS or median HNS, Agent57 lost to MuZero.

2. It is still controversial that using $\text{CHNS} \geq 100\%$ to represent the superhuman performance because it underestimates the human performance (Toromanoff et al., 2019).

3. CHNS ignores the low sample efficiency problem as other metrics using normalized scores.

In practice, CHNS can serve as an indicator to reflect whether RL agents can surpass the average human performance. The mean/median CHNS can be used to reflect the generality of the algorithms.

H.2.3. Human World Records Baseline

As (Toromanoff et al., 2019) put it, the Human Average Score Baseline potentially underestimates human performance relative to what is possible. To better reflect the performance of the algorithm compared to the human world record, we introduced a complete human world record baseline extended from (Hafner et al., 2020; Toromanoff et al., 2019) to normalize the raw score, which is called the Human World Records Normalized Score (HWRNS), which can be calculated as follows:

$$\text{HWRNS}_{g,i} = \frac{G_{g,i} - G_{g,\text{random}}}{G_{g,\text{human world records}} - G_{g,\text{random}}} \tag{17}$$

wherein g denotes the gth game of Atari, i represents the RL algorithm, $G_{i,human}$ represents the human world records, and $G_{g,random}$ represents means the performance of a random policy. Adopting HWRNS as an evaluation metric of algorithm performance has the following advantages:

1. **Intuitive comparison with human world records.** As $\text{HNS}_{g,i} \geq 100\%$ means algorithm i have surpassed the human world records performance in game g. We can directly use HWRNS to reflect which games the RL agents have surpassed the human world records, which can be used to calculate the human world records breakthrough in Atari benchmarks.

2. **Performance across algorithms become comparable.** Like the Max-Min Scaling, the HWRNS can also make two different algorithms comparable. The value of $\text{HWRNS}_{g,i}$ represents the degree to which algorithm i has surpassed the human world records in game g.

Mean HWRNS represents the mean performance of the algorithms across the 57 Atari games. Compared to mean HNS, mean HWRNS put forward higher requirements on the algorithm. Poor performance algorithms like SimPLe (Kaiser et al., 2019) will can be directly distinguished from other algorithms. It requires the algorithms to pursue a better performance across all the games rather than concentrate on one or two of them because breaking through any human world record is a huge milestone, which puts forward significant challenges to the performance and generality of the algorithm. For example, current model-free SOTA algorithms on HNS is Agent57 (Badia et al., 2020a), which only acquires 125.92% mean HWRNS, while GDI-H^3 obtained 154.27% mean HWRNS and thereby became the new state-of-the-art.

Median HWRNS represents the median performance of the algorithms across the 57 Atari games. Compared to Median HNS, median HWRNS also puts forward higher requirements for the algorithm. For example, current SOTA RL algorithms like Muzero (Schrittwieser et al., 2020) obtain much higher median HNS over GDI-H^3 but relatively lower median HWRNS.

Capped HWRNS Capped HWRNS (also called SABER) is firstly proposed and used by (Toromanoff et al., 2019), which is calculated by $\text{SABER} = \max\{\min\{\text{HWRNS}, 2\}, 0\}$. SABER also has the same problems as CHNS, and we will not repeat them here. For more details on SABER, can see (Toromanoff et al., 2019).

H.3. Learning Efficiency

As we mentioned above, traditional SOTA algorithms typically ignore the low learning efficiency problem, which makes the data used for training continuously increasing (e.g., from 10B (Kapturowski et al., 2018) to 100B (Badia et al., 2020a)). Increasing the training volume hinders the application of reinforcement learning algorithms into the real world. In this paper, we advocate not to improve the final performance via improving the learning efficiency instead of increasing the training volume. We advocate achieving SOTA within 200M training frames for Atari. To evaluate the learning efficiency of an algorithm, we introduce three promising metrics.

H.3.1. Training Scale

As one of the commonly used metrics to reveal the learning efficiency for machine learning algorithms, training scale can also serve the purpose in RL problems. In ALE, the training scale means the scale of video frames used for training. Training frames for world modeling or planning via real-world models also need to be counted in model-based settings.

H.3.2. PLAYTIME

Playtime is a unique metric of Atari, which means the equivalent real-time gameplay (Machado et al., 2018). We can use the following formula to calculate this metric:

$$\text{Playtime (day)} = \frac{\text{Num.Frames}}{108000*2*24} \tag{18}$$

For example, 200M training frames equal to 38.5 days real-time gameplay, and 100B training frames equal to 19250 days (52.7 years) real-time gameplay (Badia et al., 2020a). As far as we know, no Atari human world record was achieved by playing a game continuously for more than 52.7 years because it is less than 52.7 years since the birth of the Atari games.

H.3.3. LEARNING EFFICIENCY

As we mentioned several times while discussing the drawbacks of the normalized score, learning efficiency has been ignored in massive SOTA algorithms. Many SOTA algorithms achieved SOTA through training with vast amounts of data, which may equal 52.7 years continuously playing for a human. In this paper, we argue it is unreasonable to rely on the increase of data to improve the algorithm's performance. Thus, we proposed the following metric to evaluate the learning efficiency of an algorithm:

$$\text{Learning Efficiency} = \frac{\text{Related Evaluation Metric}}{\text{Num.Frames}} \tag{19}$$

For example, the learning efficiency of an algorithm over means HNS is $\frac{\text{mean HNS}}{\text{Num.Frames}}$, which means the algorithms obtaining higher mean HNS via lower training frames are better than those acquiring more training data methods.

H.4. Human World Record Breakthrough

As we mentioned above, we need higher requirements to prove RL agents achieve real superhuman performance. Therefore, like the CHNS (Badia et al., 2020a), the Human World Record Breakthrough (HWRB) can serve as the metric to reveal whether the algorithm has achieved the real superhuman performance, which can be calculated by $\text{HWRB} = \sum_{i=1}^{57}(\text{HWRNS} \geq 1)$.

I. Atari Benchmark

In this section, we introduce some SOTA algorithms in the Atari Benchmarks. For more details on evaluation metrics for ALE, can see App. H. For summary of the benchmark results on those evaluation metrics can see App. J.

I.1. Model-Free Reinforcement Learning

I.1.1. RAINBOW

Rainbow (Hessel et al., 2017) is a classic value-based RL algorithm among the DQN algorithm family, which has fruitfully combined six extensions of the DQN algorithm family. It is recognized to achieve state-of-the-art performance on the ALE benchmark. Thus, we select it as one of the representative algorithms of the SOTA DQN algorithms.

I.1.2. IMPALA

IMPALA, namely the Importance Weighted Actor Learner Architecture (Espeholt et al., 2018), is a classic distributed off-policy actor-critic framework, which decouples acting from learning and learning from experience trajectories using V-trace. IMPALA actors communicate trajectories of experience (sequences of states, actions, and rewards) to a centralized learner, which boosts distributed large-scale training. Thus, we select it as one of the representative algorithms of the traditional distributed RL algorithm.

I.1.3. LASER

LASER (Schmitt et al., 2020) is a classic Actor-Critic algorithm, which investigated the combination of Actor-Critic algorithms with a uniform large-scale experience replay. It trained populations of actors with shared experiences and claimed to achieve SOTA in Atari. Thus, we select it as one of the SOTA RL algorithms within 200M training frames.

I.1.4. R2D2

(Kapturowski et al., 2018) Like IMPALA, R2D2 (Kapturowski et al., 2018) is also a classic distributed RL algorithms. It trained RNN-based RL agents from distributed prioritized experience replay, which achieved SOTA in Atari. Thus, we select it as one of the representative value-based distributed RL algorithms.

I.1.5. NGU

One of the classical problems in ALE for RL agents is the hard exploration problems (Ecoffet et al., 2019; Bellemare et al., 2013; Badia et al., 2020a) like *Private Eye, Montezuma's Revenge, Pitfall!*. NGU (Badia et al., 2020b), or Never Give Up, try to ease this problem by augmenting the reward signal with an internally generated intrinsic reward that is sensitive to novelty at two levels: short-term novelty within an episode and long-term novelty across episodes. It then learns a family of policies for exploring and exploiting (sharing the same parameters) to obtain the highest score under the exploitative policy. NGU has achieved SOTA in Atari and thus we selected it as one of the representative population-based model-free RL algorithms.

I.1.6. AGENT57

Agent57 (Badia et al., 2020a) is the SOTA model-free RL algorithms on CHNS or Median HNS of Atari Benchmark. Built on the NGU agents, Agent57 proposed a novel state-action value function parameterization method and adopted an adaptive exploration over a family of policies, which overcome the drawback of NGU (Badia et al., 2020a). We select it as one of the SOTA model-free RL algorithms.

I.1.7. GDI

GDI, or Generalized Data Distribution Iteration, claimed to have achieved SOTA on mean/median HWRNS, mean HNS, HWRB, median SABER of Atari Benchmark. GDI is one of the novel Reinforcement Learning paradigms, which combined a data distribution optimization operator into the traditional generalized policy iteration (GPI) (Sutton & Barto, 2018) and thus achieved human-level learning efficiency. Thus, we select them as one of the SOTA model-free RL algorithms.

I.2. Model-Based Reinforcement Learning

I.2.1. SimPLe

As one of the classic model-based RL algorithms on Atari, SimPLe, or Simulated Policy Learning (Kaiser et al., 2019), adopted a video prediction model to enable RL agents to solve Atari problems with higher sample efficiency. It claimed to outperform the SOTA model-free algorithms in most games, so we selected it as representative model-based RL algorithms.

I.2.2. Dreamer-V2

Dreamer-V2 (Hafner et al., 2020) built world models to facilitate generalization across the experience and allow learning behaviors from imagined outcomes in the compact latent space of the world model to increase sample efficiency. Dreamer-V2 is claimed to achieve SOTA in Atari and thus we select it as one of the SOTA model-based RL algorithms within the 200M training scale.

I.2.3. Muzero

Muzero (Schrittwieser et al., 2020) combined a tree-based search with a learned model and has achieved superhuman performance on Atari. We thus selected it as one of the SOTA model-based RL algorithms.

I.3. Other SOTA algorithms

I.3.1. Go-Explore

As mentioned in NGU, a grand challenge in reinforcement learning is intelligent exploration, which is called the hard-exploration problem (Machado et al., 2018). Go-Explore (Ecoffet et al., 2019) adopted three principles to solve this problem. Firstly, agents remember previously visited states. Secondly, agents first return to a promising state and then explore it. Finally, solve simulated environment through any available means, and then robustify via imitation learning. Go-Explore has achieved SOTA in Atari, so we select it as one of the SOTA algorithms of the hard exploration problem.

I.3.2. Muesli

Muesli (Hessel et al., 2021) proposed a novel policy update that combines regularized policy optimization with model learning as an auxiliary loss. It acts directly with a policy network and has a computation speed comparable to model-free baselines. As it claimed to achieve SOTA in Atari within 200M training frames, we select it as one of the SOTA RL algorithms within 200M training frames.

I.4. Summary of Benchmark Results

This part summarizes the results among all the algorithms we mentioned above on the human world record benchmark for Atari. In Figs, we illustrated the benchmark results on HNS, HWRNS, SABER, and the corresponding training scale. 6, 9 and 12, HWRB and corresponding game time and learning efficiency in Fig. 13. From those results, we see GDI has achieved SOTA in learning efficiency, HWRB, HWRNS, mean HNS, and median SABER within 200M training frames. Agent57 has achieved SOTA in mean SABER, and Muzero (Schrittwieser et al., 2020) has achieved SOTA in median HNS.

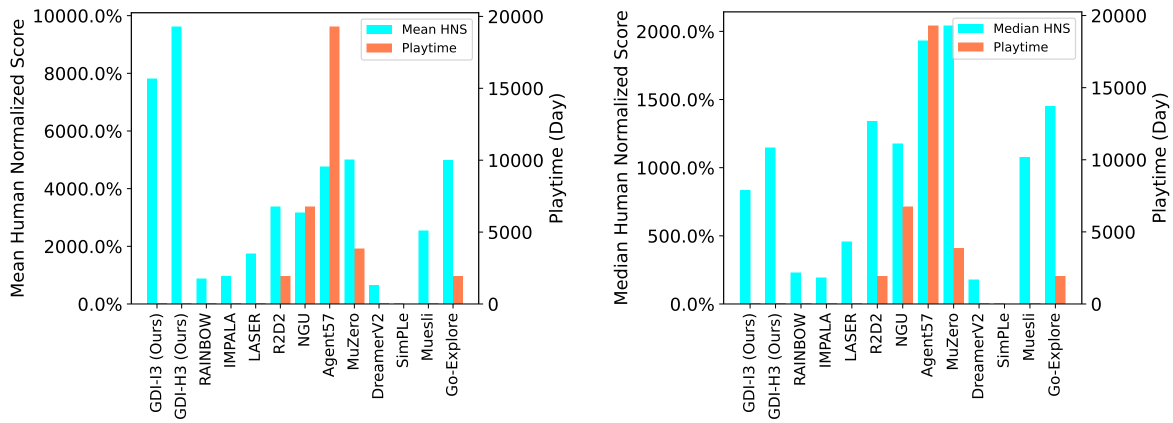


Figure 4. SOTA algorithms of Atari 57 games on mean and median HNS (%) and playtime.

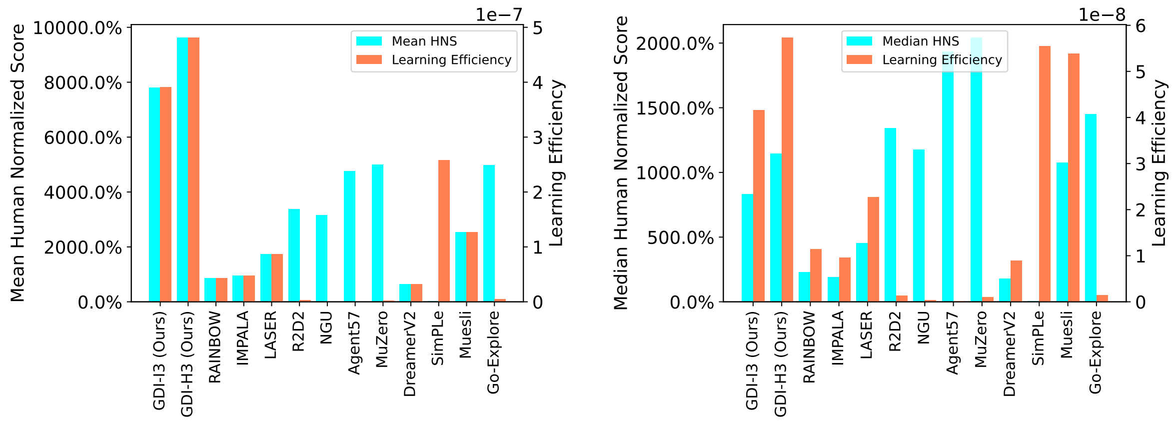


Figure 5. SOTA algorithms of Atari 57 games on mean and median HNS (%) and corresponding learning efficiency calculated by $\frac{\text{MEAN HNS/MEDIAN HNS}}{\text{TRAINING FRAMES}}$.

J. Summary of Benchmark Results

In this section, we illustrate the benchmark results of all the SOTA algorithms mentioned in this paper. For more details on these algorithms, can see App. I.

J.1. RL Benchmarks on HNS

We report several milestones of Atari benchmarks on HNS, including DQN (Mnih et al., 2015), RAINBOW (Hessel et al., 2017), IMPALA (Espeholt et al., 2018), LASER (Schmitt et al., 2020), R2D2 (Kapturowski et al., 2018), NGU (Badia et al., 2020b), Agent57 (Badia et al., 2020a), Go-Explore (Ecoffet et al., 2019), MuZero (Schrittwieser et al., 2020), DreamerV2 (Hafner et al., 2020), SimPLe (Kaiser et al., 2019) and Muesli (Hessel et al., 2021). We summarize mean HNS and median HNS of these algorithms with their playtime (human playtime), learning efficiency , and training scale in Fig 4, 5 and 6.

J.2. RL Benchmarks on HWRNS

We report several milestones of Atari benchmarks on Human World Records Normalized Score (HWRNS), including DQN (Mnih et al., 2015), RAINBOW (Hessel et al., 2017), IMPALA (Espeholt et al., 2018), LASER (Schmitt et al., 2020), R2D2 (Kapturowski et al., 2018), NGU (Badia et al., 2020b), Agent57 (Badia et al., 2020a), Go-Explore (Ecoffet et al., 2019), MuZero (Schrittwieser et al., 2020), DreamerV2 (Hafner et al., 2020), SimPLe (Kaiser et al., 2019) and Muesli (Hessel

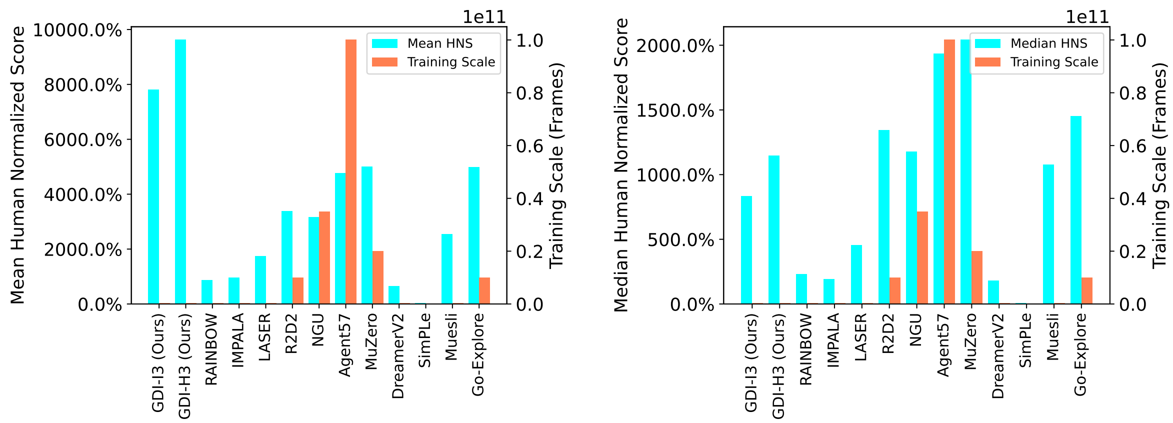


Figure 6. SOTA algorithms of Atari 57 games on mean and median HNS (%) and corresponding training scale.

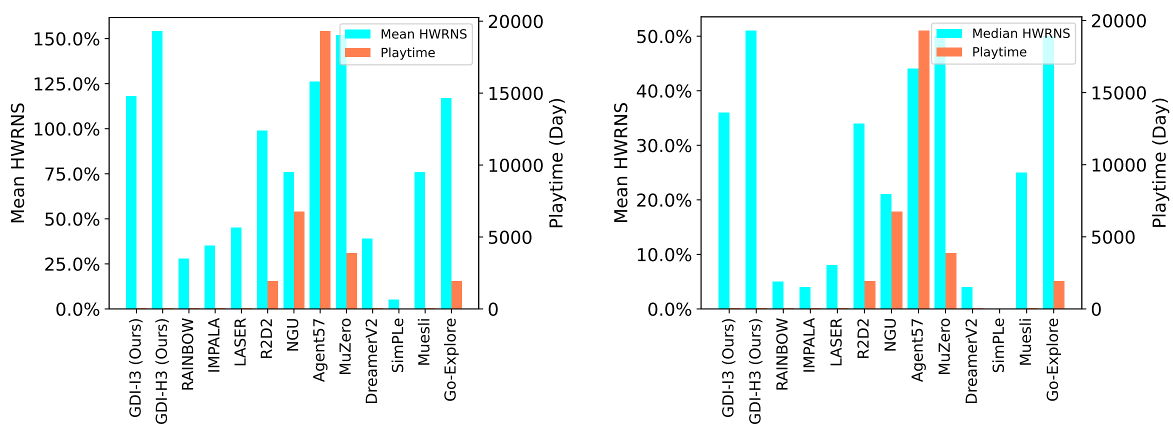


Figure 7. SOTA algorithms of Atari 57 games on mean and median HWRNS (%) and corresponding playtime.

et al., 2021). We summarize mean HWRNS and median HWRNS of these algorithms with their playtime (day), learning efficiency , and training scale in Fig 7, 8 and 9.

J.3. RL Benchmarks on SABER

We report several milestones of Atari benchmarks on Standardized Atari BEnchmark for RL (SABER), including DQN (Mnih et al., 2015), RAINBOW (Hessel et al., 2017), IMPALA (Espeholt et al., 2018), LASER (Schmitt et al., 2020), R2D2 (Kapturowski et al., 2018), NGU (Badia et al., 2020b), Agent57 (Badia et al., 2020a), Go-Explore (Ecoffet et al., 2019), MuZero (Schrittwieser et al., 2020), DreamerV2 (Hafner et al., 2020), SimPLe (Kaiser et al., 2019) and Muesli (Hessel et al., 2021). We summarize mean SABER and median SABER of these algorithms with their playtime, learning efficiency, and training scale in Figs 10, 11 and 12.

J.4. RL Benchmarks on HWRB

We report several milestones of Atari benchmarks on HWRB, including DQN (Mnih et al., 2015), RAINBOW (Hessel et al., 2017), IMPALA (Espeholt et al., 2018), LASER (Schmitt et al., 2020), R2D2 (Kapturowski et al., 2018), NGU (Badia et al., 2020b), Agent57 (Badia et al., 2020a), Go-Explore (Ecoffet et al., 2019), MuZero (Schrittwieser et al., 2020), DreamerV2 (Hafner et al., 2020), SimPLe (Kaiser et al., 2019) and Muesli (Hessel et al., 2021). We summarize HWRB of these algorithms with their playtime, learning efficiency , and training scale in Figs 13.

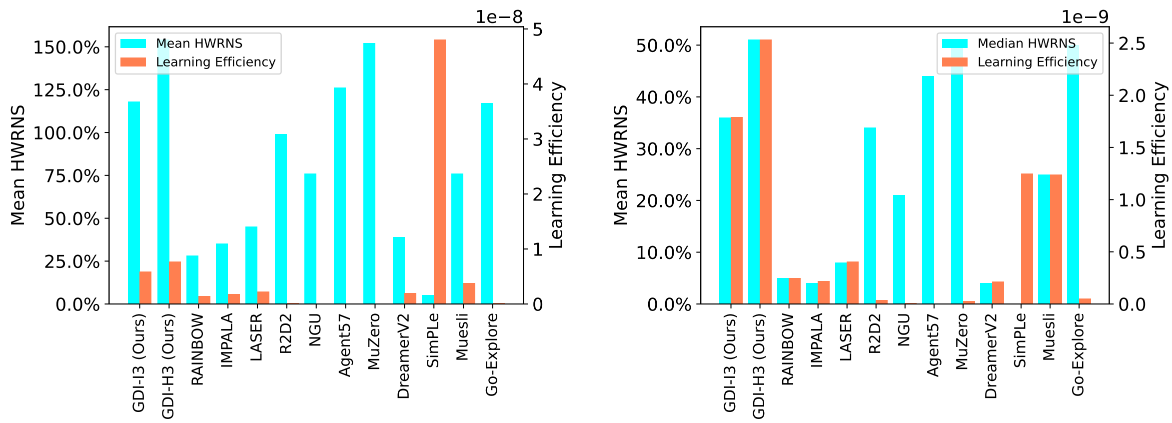


Figure 8. SOTA algorithms of Atari 57 games on mean and median HWRNS (%) and corresponding learning efficiency calculated by $\frac{\text{MEAN HWRNS/MEDIAN HWRNS}}{\text{TRAINING FRAMES}}$.

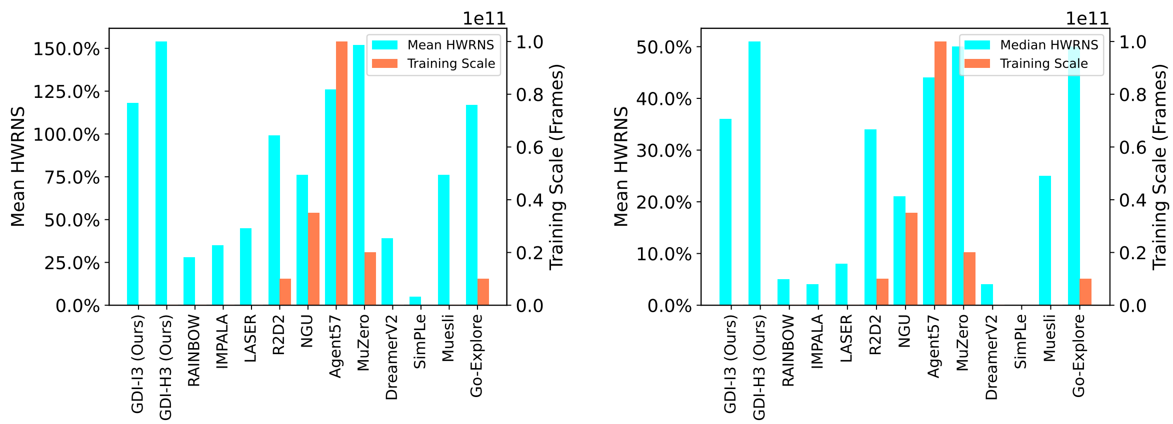


Figure 9. SOTA algorithms of Atari 57 games on mean and median HWRNS (%) and corresponding training scale.

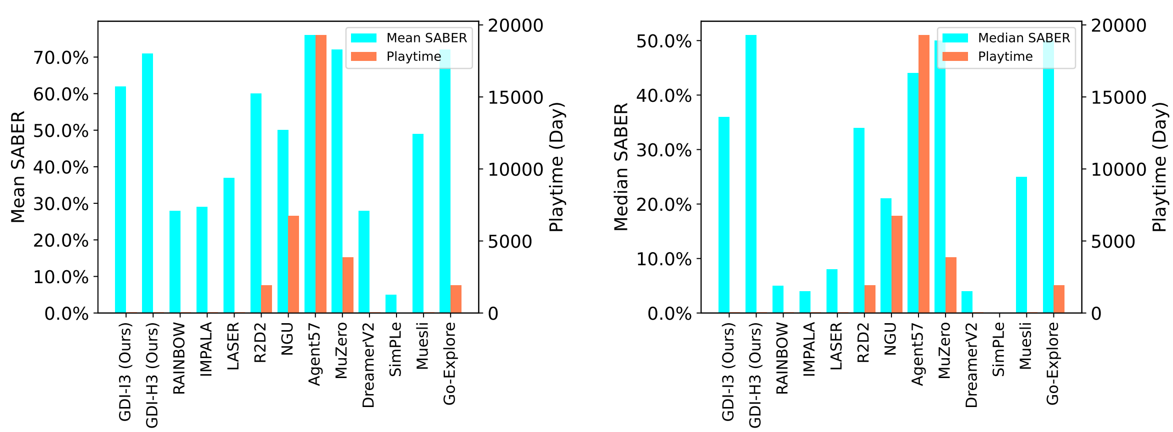


Figure 10. SOTA algorithms of Atari 57 games on mean and median SABER (%) and corresponding playtime.

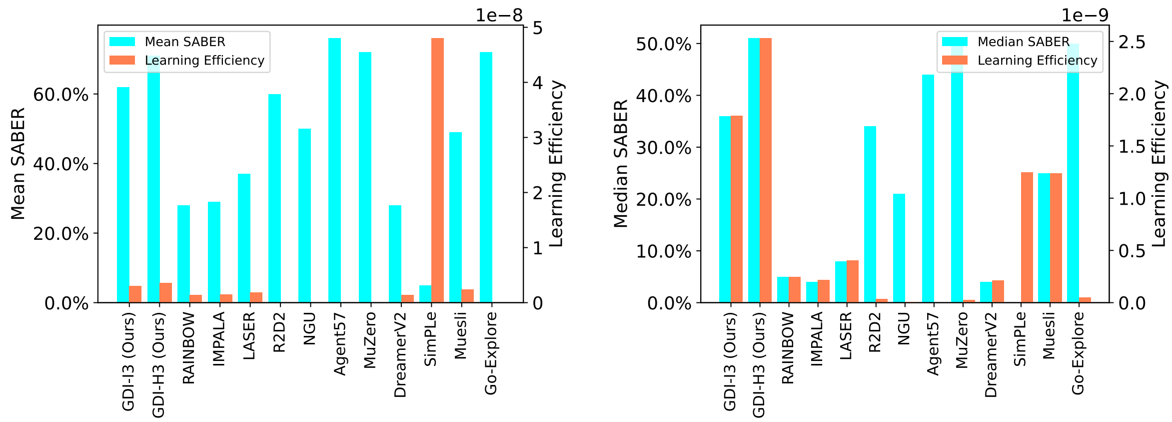


Figure 11. SOTA algorithms of Atari 57 games on mean and median SABER (%) and corresponding learning efficiency calculated by $\frac{\text{MEAN SABER/MEDIAN SABER}}{\text{TRAINING FRAMES}}$.

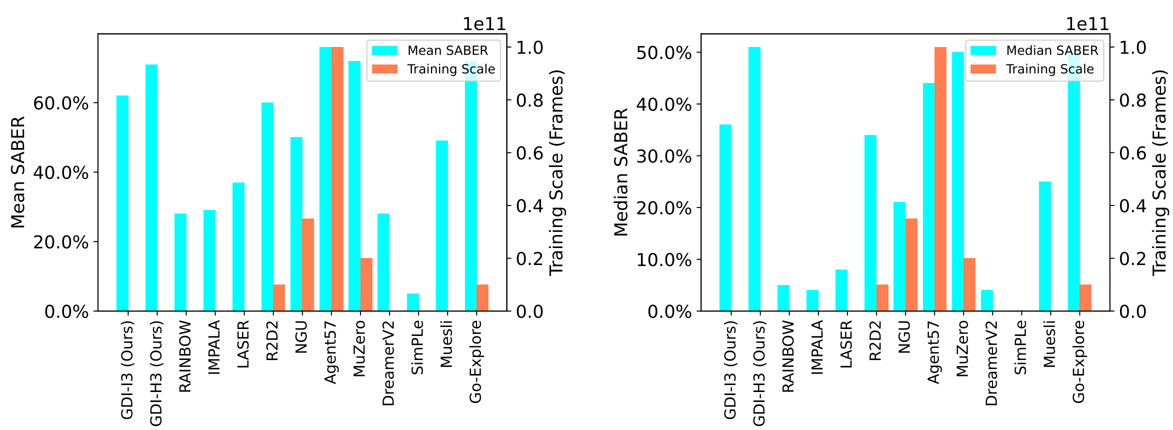


Figure 12. SOTA algorithms of Atari 57 games on mean and median SABER (%) and corresponding training scale.

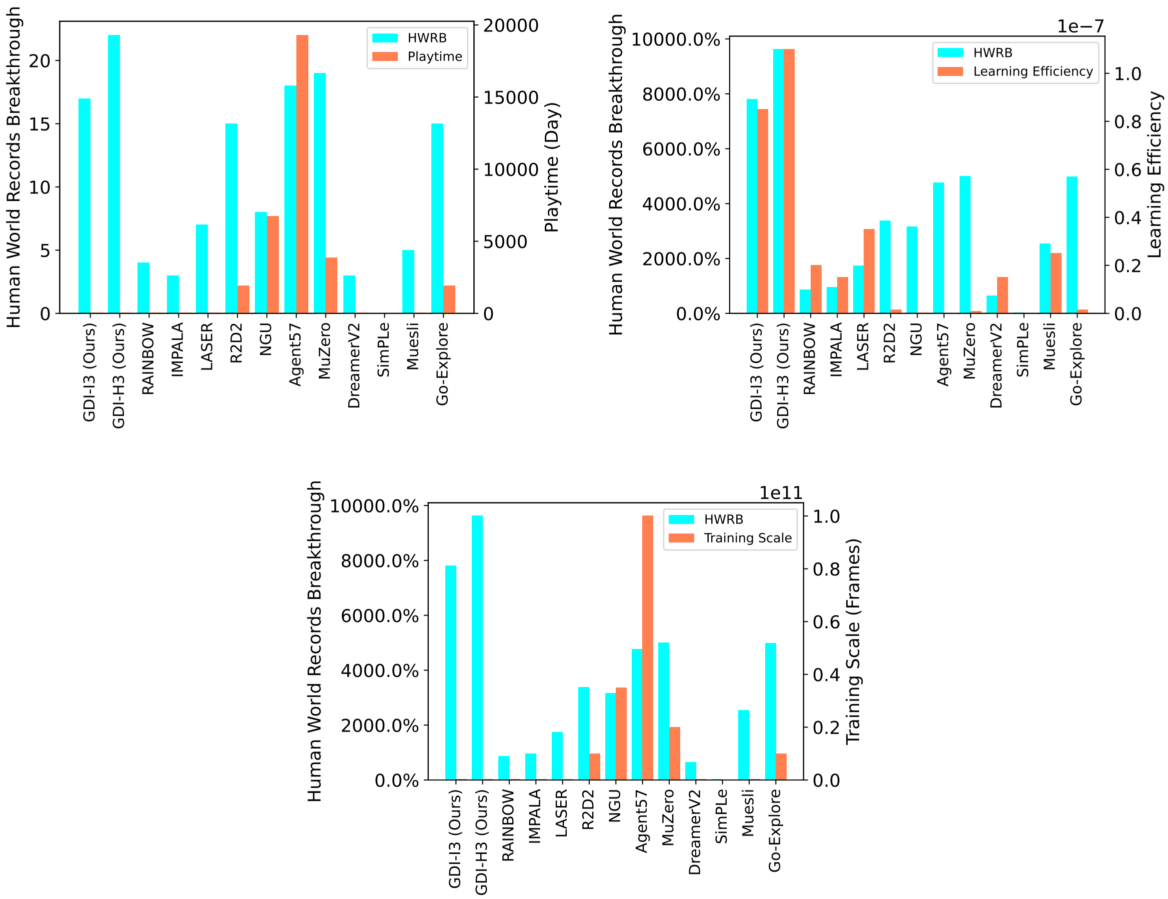


Figure 13. SOTA algorithms of Atari 57 games on HWRB. HWRB of SimPLe is 0, so it's not shown in the up-right figure.

K. Experimental Results

In this section, we report the performance of GDI-H^3, GDI-I^3, and many well-known SOTA algorithms, including both the model-based and model-free methods (see App. I). First of all, we summarize the performance of all the algorithms over all the evaluation criteria of our evaluation system in App. K.1 which is mentioned in App. F. In the following three parts, we visualize the performance of GDI-H^3, GDI-I^3 over HNS in App. K.2, HWRNS in App. K.3, SABER in App. K.4 via histogram. Furthermore, we detail all the original scores of all the algorithms and provide raw data that calculates those evaluation criteria, wherein we first provide all the human world records in 57 Atari games and calculate the HNS in App. K.5, HWRNS in App. K.6 and SABER in App. K.7 of all 57 Atari games. We further provide all the evaluation curves of GDI-H^3 and GDI-I^3 over 57 Atari games in the App. K.8.

K.1. Full Performance Comparison

In this part, we summarize the performance of all mentioned algorithms over all the evaluation criteria in Tab. 6. In the following sections, we will detail the performance of each algorithm on all Atari games one by one.

Table 6. Full performance comparison on Atari. The units of training scale is sampled frames. The units of playtime is huamn playtime (day). HNS(%), HWRNS(%), and SABER(%) adopts the percentage format (i.e., %). Bold scores indicate the SOTA performance

Algorithms	Training Scale	Playtime	HWRB	Mean HNS	Median HNS	Mean HWRNS	Median HWRNS	Mean SABER	Median SABER
GDI-I^3	2.00E+08	38.5	**17**	**7810.1**	**832.5**	**117.98**	**35.78**	**61.66**	**35.78**
Rainbow	2.00E+08	38.5	4	873.54	230.99	28.39	4.92	28.39	4.92
IMPALA	2.00E+08	38.5	3	956.99	191.82	34.52	4.31	29.45	4.31
LASER	2.00E+08	38.5	7	1740.94	454.91	45.39	8.08	36.78	8.08
GDI-I^3	2.00E+08	38.5	17	**7810.1**	832.5	117.98	35.78	61.66	35.78
R2D2	1.00E+10	1929	15	3373.48	1342.27	98.78	33.62	60.43	33.62
NGU	3.50E+10	6751.5	8	3169.07	1174.92	76.00	21.19	50.47	21.19
Agent57	1.00E+11	19290	**18**	4762.17	**1933.49**	**125.92**	**43.62**	**76.26**	**43.62**
GDI-I^3	2.00E+08	38.5	17	**7810.1**	832.5	117.98	35.78	61.66	35.78
SimPLe	1.00E+06	0.19	0	25.78	5.55	4.80	0.13	4.80	0.13
DreamerV2	2.00E+08	38.58	3	642.49	178.04	38.60	4.29	27.73	4.29
MuZero	2.00E+10	3858	**19**	4994.97	**2041.12**	**152.10**	**49.80**	**71.94**	**49.80**
GDI-I^3	2.00E+08	38.5	**17**	**7810.1**	832.5	117.98	35.78	61.66	35.78
Muesli	2.00E+08	38.5	5	2538.12	1077.47	75.52	24.86	48.74	24.86
Go-Explore	1.00E+10	1929	15	4989.31	**1451.55**	116.89	**50.50**	**71.80**	**50.50**
GDI-H^3	2.00E+08	38.5	**22**	**9620.33**	**1146.39**	**154.27**	**50.63**	**71.26**	**50.63**
Rainbow	2.00E+08	38.5	4	873.54	230.99	28.39	4.92	28.39	4.92
IMPALA	2.00E+08	38.5	3	956.99	191.82	34.52	4.31	29.45	4.31
LASER	2.00E+08	38.5	7	1740.94	454.91	45.39	8.08	36.78	8.08
GDI-H^3	2.00E+08	38.5	**22**	**9620.33**	1146.39	**154.27**	**50.63**	71.26	**50.63**
R2D2	1.00E+10	1929	15	3373.48	1342.27	98.78	33.62	60.43	33.62
NGU	3.50E+10	6751.5	8	3169.07	1174.92	76.00	21.19	50.47	21.19
Agent57	1.00E+11	19290	18	4762.17	**1933.49**	125.92	43.62	**76.26**	43.62
GDI-H^3	2.00E+08	38.5	**22**	**9620.33**	1146.39	**154.27**	**50.63**	71.26	**50.63**
SimPLe	1.00E+06	0.19	0	25.78	5.55	4.80	0.13	4.80	0.13
DreamerV2	2.00E+08	38.58	3	642.49	178.04	38.60	4.29	27.73	4.29
MuZero	2.00E+10	3858	19	4994.97	**2041.12**	152.10	49.80	**71.94**	49.80
GDI-H^3	2.00E+08	38.5	**22**	**9620.33**	1146.39	**154.27**	**50.63**	71.26	**50.63**
Muesli	2.00E+08	38.5	5	2538.12	1077.47	75.52	24.86	48.74	24.86
Go-Explore	1.00E+10	1929	15	4989.31	**1451.55**	116.89	50.50	**71.80**	50.50

K.2. Figure of HNS

In this part, we visualize the HNS using GDI-H^3 and GDI-I^3 in all 57 games. The HNS histogram of GDI-I^3 is illustrated in Fig. 14. The HNS histogram of GDI-H^3 is illustrated in Fig. 15.

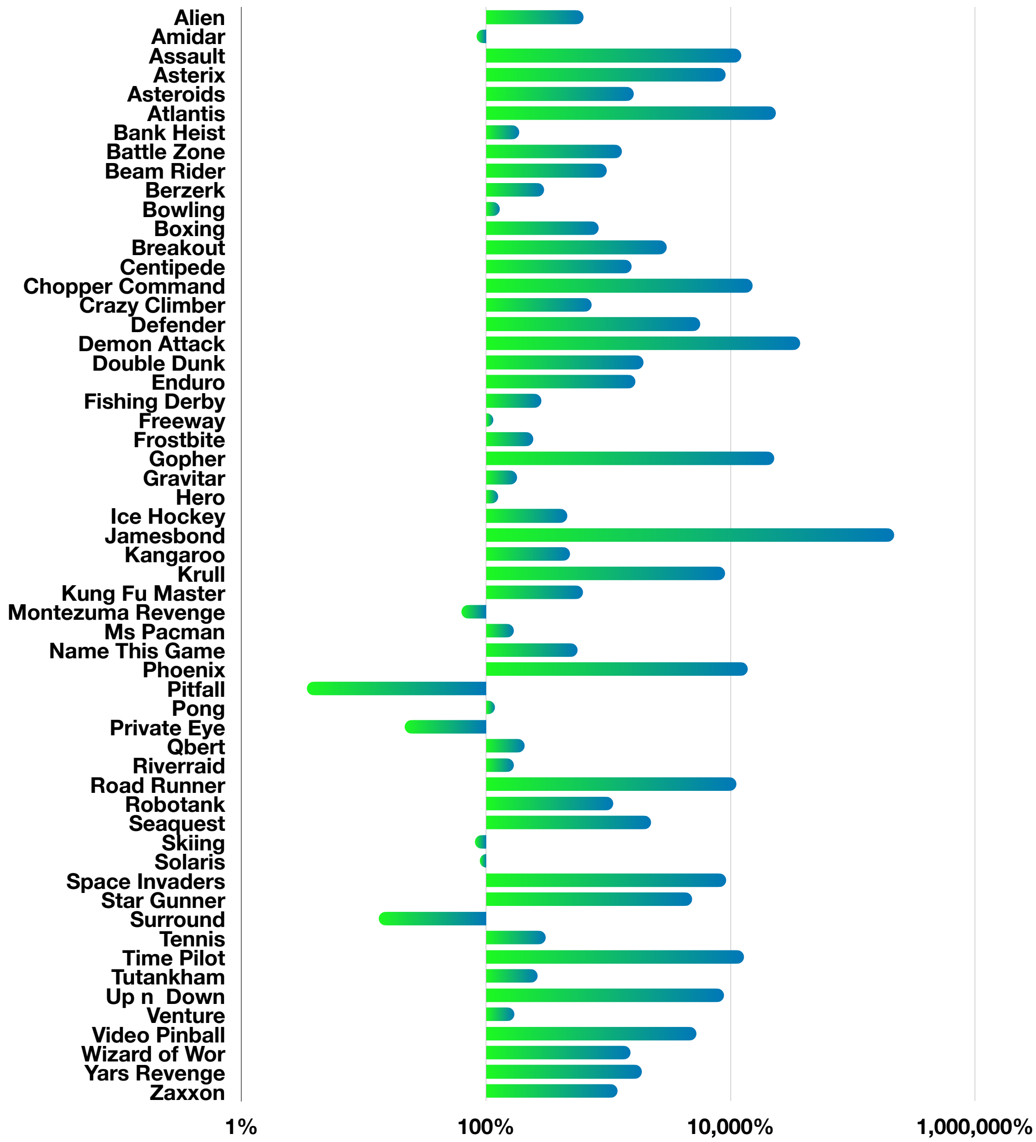


Figure 14. HNS (%) of Atari 57 games using GDI-I^3.

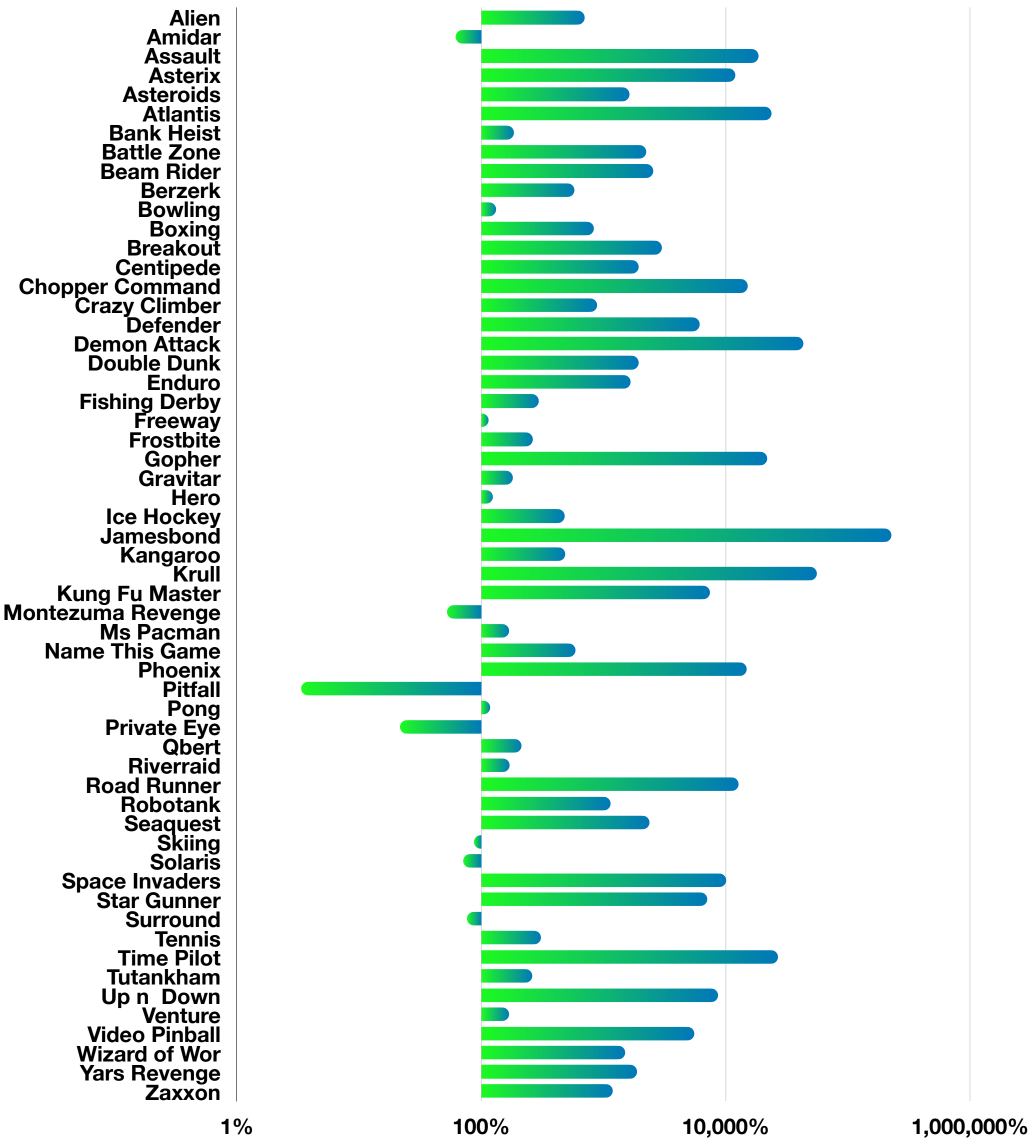


Figure 15. HNS (%) of Atari 57 games using GDI-H^3.

K.3. Figure of HWRNS

In this part, we visualize the HWRNS (Hafner et al., 2020; Toromanoff et al., 2019) using GDI-H^3 and GDI-I^3 in all 57 games. The HWRNS histogram of GDI-I^3 is illustrated in Fig. 16. The HWRNS histogram of GDI-H^3 is illustrated in Fig. 17.

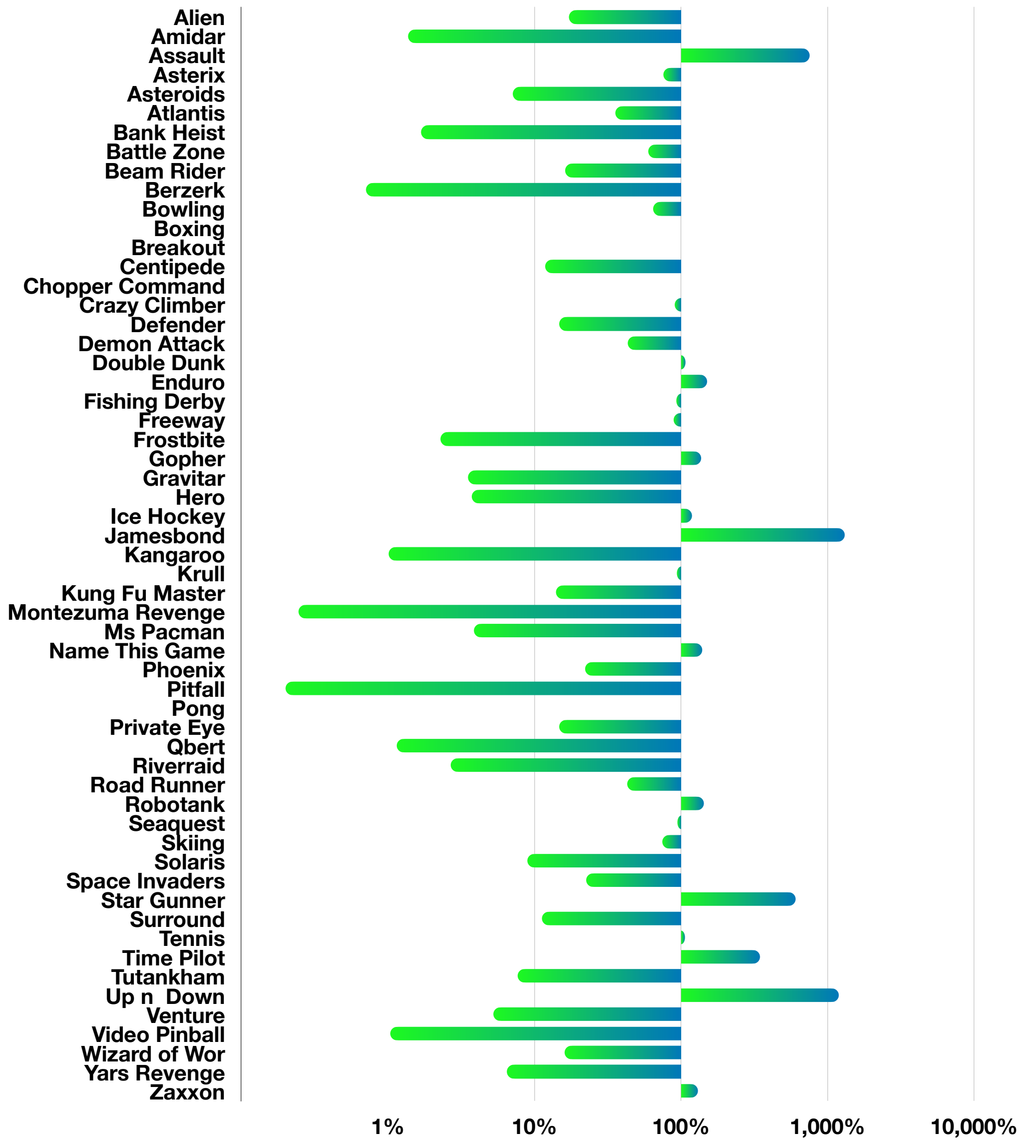


Figure 16. HWRNS (%) of Atari 57 games using GDI-I^3.

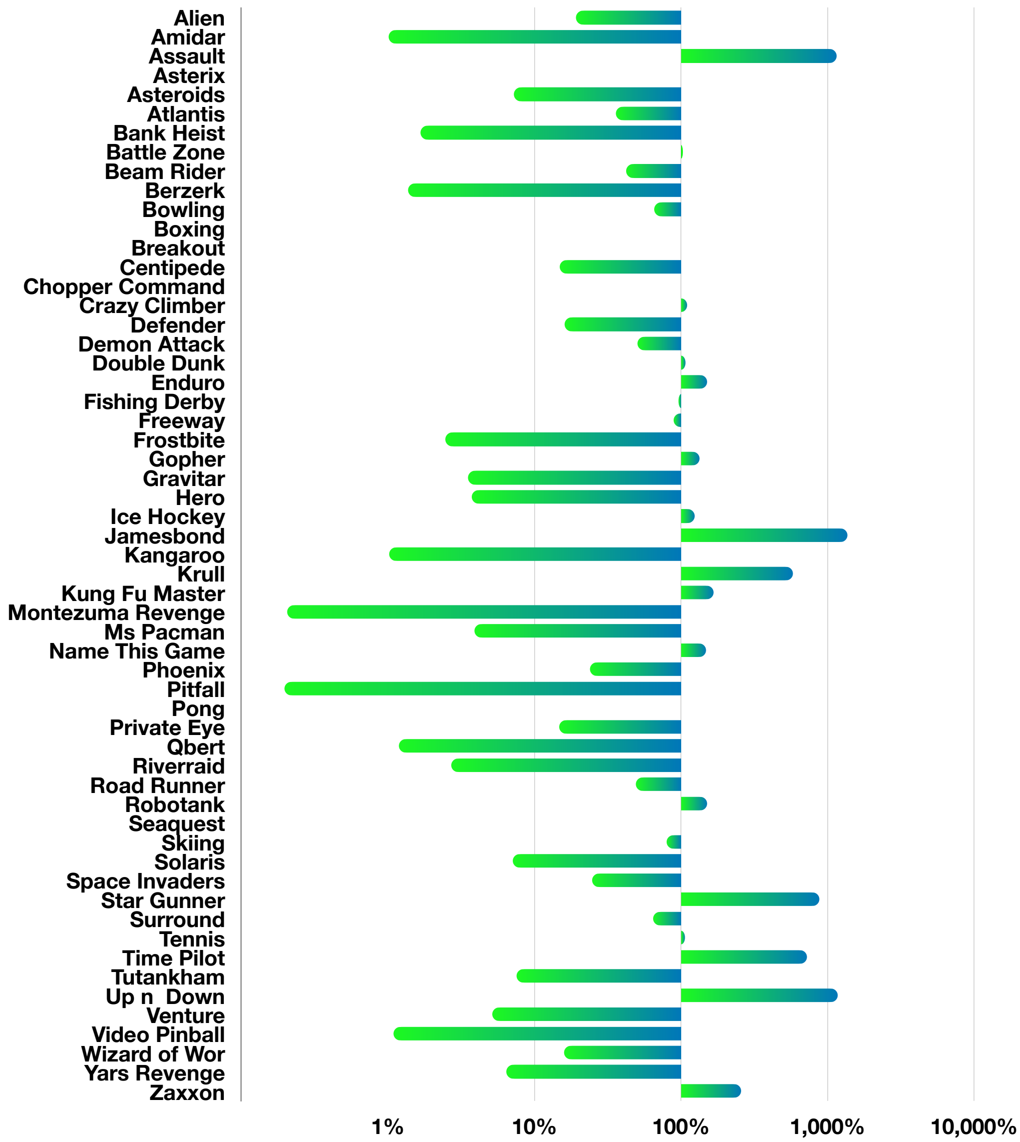


Figure 17. HWRNS (%) of Atari 57 games using GDI-H^3.

K.4. Figure of SABER

In this part, we illustrate the SABER (Hafner et al., 2020; Toromanoff et al., 2019) using GDI-H^3 and GDI-I^3 in all 57 games. The SABER histogram of GDI-I^3 is illustrated in Fig. 18. The SABER histogram of GDI-H^3 is illustrated in Fig. 19.

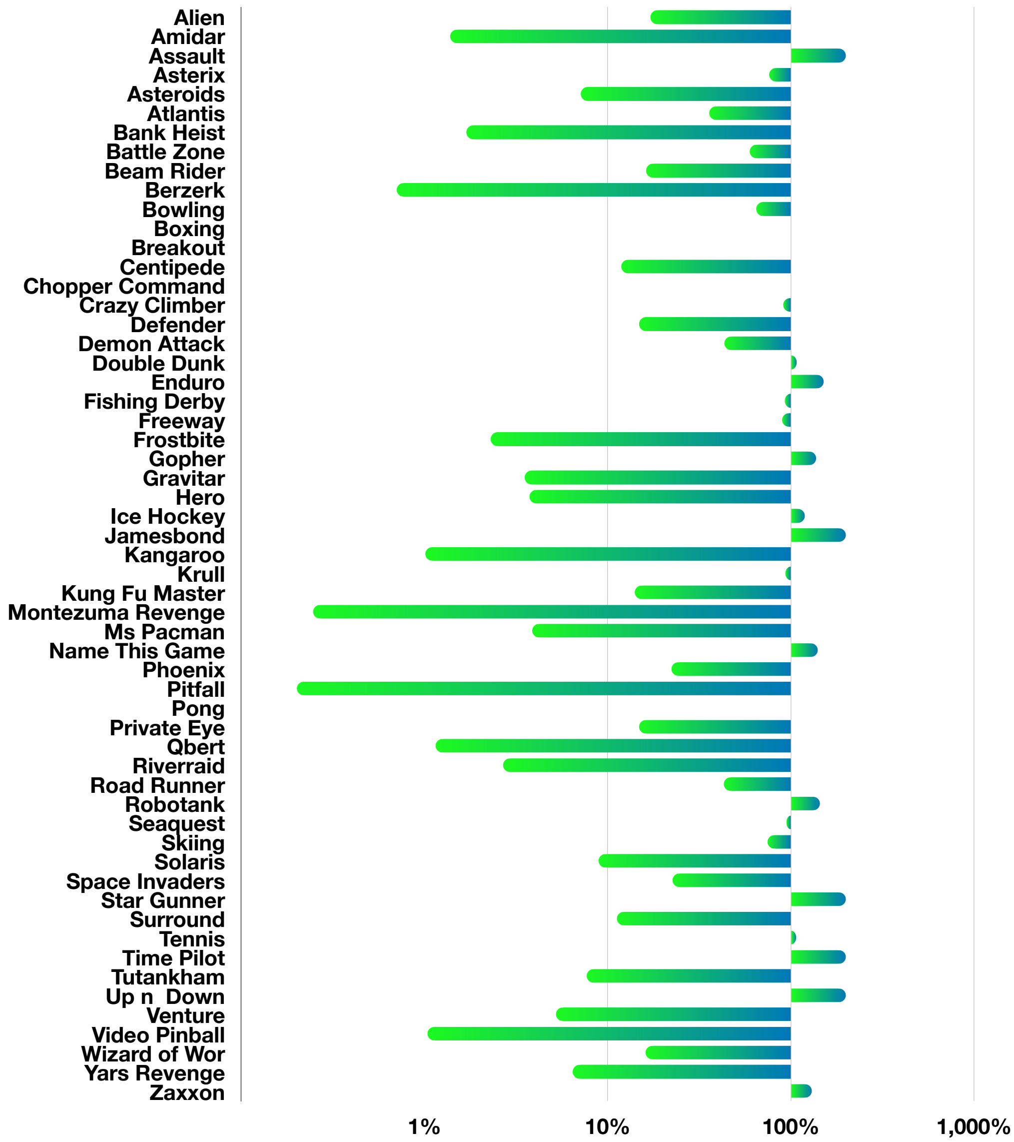


Figure 18. SABER (%) of Atari 57 games using GDI-I^3.

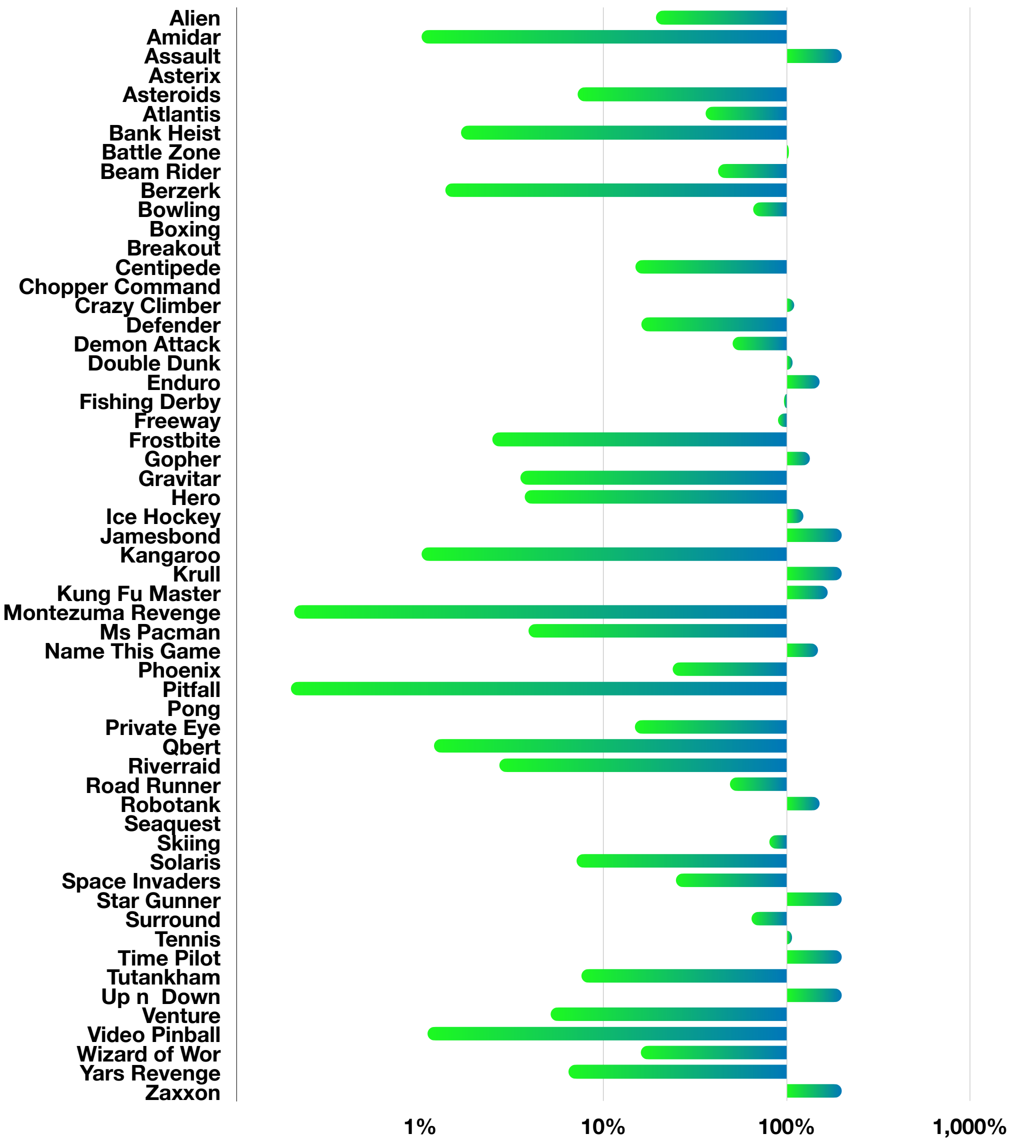


Figure 19. SABER (%) of Atari 57 games using GDI-H^3.

K.5. Atari Games Table of Scores Based on Human Average Records

In this part, we detail the raw score of several representative SOTA algorithms , including the SOTA 200M model-free algorithms, SOTA 10B+ model-free algorithms, SOTA model-based algorithms and other SOTA algorithms.[1] Additionally, we calculate the Human Normalized Score (HNS) of each game with each algorithm. First of all, we demonstrate the sources of the scores that we used. Random scores and average human's scores are from (Badia et al., 2020a). Rainbow's scores are from (Hessel et al., 2017). IMPALA's scores are from (Espeholt et al., 2018). LASER's scores are from (Schmitt et al., 2020), with no sweep at 200M. As there are many versions of R2D2 and NGU, we use original papers'. R2D2's scores are from (Kapturowski et al., 2018). NGU's scores are from (Badia et al., 2020b). Agent57's scores are from (Badia et al., 2020a). MuZero's scores are from (Schrittwieser et al., 2020). DreamerV2's scores are from (Hafner et al., 2020). SimPLe's scores are from (Kaiser et al., 2019). Go-Explore's scores are from (Ecoffet et al., 2019). Muesli's scores are from (Hessel et al., 2021). In the following, we detail the raw scores and HNS of each algorithm on 57 Atari games.

[1] 200M and 10B+ represent the training scale.

Table 7. Score table of SOTA 200M model-free algorithms on HNS(%) (GDI-I^3).

Games	RND	HUMAN	RAINBOW	HNS	IMPALA	HNS	LASER	HNS	GDI-I^3	HNS
Scale			200M		200M		200M		200M	
Alien	227.8	7127.8	9491.7	134.26	15962.1	228.03	35565.9	512.15	43384	625.45
Amidar	5.8	1719.5	**5131.2**	**299.08**	1554.79	90.39	1829.2	106.4	1442	83.81
Assault	222.4	742	14198.5	2689.78	19148.47	3642.43	21560.4	4106.62	63876	12250.50
Asterix	210	8503.3	428200	5160.67	300732	3623.67	240090	2892.46	759910	9160.41
Asteroids	719	47388.7	2712.8	4.27	108590.05	231.14	213025	454.91	751970	1609.72
Atlantis	12850	29028.1	826660	5030.32	849967.5	5174.39	841200	5120.19	3803000	23427.66
Bank Heist	14.2	753.1	1358	181.86	1223.15	163.61	569.4	75.14	**1401**	**187.68**
Battle Zone	236	37187.5	62010	167.18	20885	55.88	64953.3	175.14	478830	1295.20
Beam Rider	363.9	16926.5	16850.2	99.54	32463.47	193.81	90881.6	546.52	162100	976.51
Berzerk	123.7	2630.4	2545.6	96.62	1852.7	68.98	**25579.5**	**1015.51**	7607	298.53
Bowling	23.1	160.7	30	5.01	59.92	26.76	48.3	18.31	201.9	129.94
Boxing	0.1	12.1	99.6	829.17	99.96	832.17	**100**	**832.5**	**100**	**832.50**
Breakout	1.7	30.5	417.5	1443.75	787.34	2727.92	747.9	2590.97	**864**	**2994.10**
Centipede	2090.9	12017	8167.3	61.22	11049.75	90.26	**292792**	**2928.65**	155830	1548.84
Chopper Command	811	7387.8	16654	240.89	28255	417.29	761699	11569.27	**999999**	**15192.62**
Crazy Climber	10780.5	36829.4	168788.5	630.80	136950	503.69	167820	626.93	201000	759.39
Defender	2874.5	18688.9	55105	330.27	185203	1152.93	336953	2112.50	893110	5629.27
Demon Attack	152.1	1971	111185	6104.40	132826.98	7294.24	133530	7332.89	675530	37131.12
Double Dunk	-18.6	-16.4	-0.3	831.82	-0.33	830.45	14	1481.82	**24**	**1936.36**
Enduro	0	860.5	2125.9	247.05	0	0.00	0	0.00	**14330**	**1665.31**
Fishing Derby	-91.7	-38.8	31.3	232.51	44.85	258.13	45.2	258.79	59	285.71
Freeway	0	29.6	**34**	**114.86**	0	0.00	0	0.00	**34**	**114.86**
Frostbite	65.2	4334.7	9590.5	223.10	317.75	5.92	5083.5	117.54	10485	244.05
Gopher	257.6	2412.5	70354.6	3252.91	66782.3	3087.14	114820.7	5316.40	**488830**	**22672.63**
Gravitar	173	3351.4	1419.3	39.21	359.5	5.87	1106.2	29.36	5905	180.34
Hero	1027	30826.4	**55887.4**	**184.10**	33730.55	109.75	31628.7	102.69	38330	125.18
Ice Hockey	-11.2	0.9	1.1	101.65	3.48	121.32	17.4	236.36	44.94	463.97
Jamesbond	29	302.8	19809	72.24	601.5	209.09	37999.8	13868.08	594500	217118.70
Kangaroo	52	3035	**14637.5**	**488.05**	1632	52.97	14308	477.91	14500	484.34
Krull	1598	2665.5	8741.5	669.18	8147.4	613.53	9387.5	729.70	97575	8990.82
Kung Fu Master	258.5	22736.3	52181	230.99	43375.5	191.82	607443	2701.26	140440	623.64
Montezuma Revenge	0	**4753.3**	384	8.08	0	0.00	0.3	0.01	3000	63.11
Ms Pacman	307.3	6951.6	5380.4	76.35	7342.32	105.88	6565.5	94.19	11536	169.00
Name This Game	2292.3	8049	13136	188.37	21537.2	334.30	26219.5	415.64	34434	558.34
Phoenix	761.5	7242.6	108529	1662.80	210996.45	3243.82	519304	8000.84	894460	13789.30
Pitfall	-229.4	**6463.7**	0	3.43	-1.66	3.40	-0.6	3.42	0	3.43
Pong	-20.7	14.6	20.9	117.85	20.98	118.07	**21**	**118.13**	**21**	**118.13**
Private Eye	24.9	**69571.3**	4234	6.05	98.5	0.11	96.3	0.10	15100	21.68
Qbert	163.9	13455.0	33817.5	253.20	**351200.12**	**2641.14**	21449.6	160.15	27800	207.93
Riverraid	1338.5	17118.0	22920.8	136.77	29608.05	179.15	**40362.7**	**247.31**	28075	169.44
Road Runner	11.5	7845	62041	791.85	57121	729.04	45289	578.00	878600	11215.78
Robotank	2.2	11.9	61.4	610.31	12.96	110.93	62.1	617.53	108.2	1092.78
Seaquest	68.4	42054.7	15898.9	37.70	1753.2	4.01	2890.3	6.72	943910	2247.98
Skiing	-17098	**-4336.9**	-12957.8	32.44	-10180.38	54.21	-29968.4	-100.86	-6774	80.90
Solaris	1236.3	**12326.7**	3560.3	20.96	2365	10.18	2273.5	9.35	11074	88.70
Space Invaders	148	1668.7	18789	1225.82	43595.78	2857.09	51037.4	3346.45	140460	9226.80
Star Gunner	664	10250	127029	1318.22	200625	2085.97	321528	3347.21	465750	4851.72
Surround	-10	6.5	**9.7**	**119.39**	7.56	106.42	8.4	111.52	-7.8	13.33
Tennis	-23.8	-8.3	0	153.55	0.55	157.10	12.2	232.26	**24**	**308.39**
Time Pilot	3568	5229.2	12926	563.36	48481.5	2703.84	105316	6125.34	216770	12834.99
Tutankham	11.4	167.6	241	146.99	292.11	179.71	278.9	171.25	**423.9**	**264.08**
Up N Down	533.4	11693.2	125755	1122.08	332546.75	2975.08	345727	3093.19	**986440**	**8834.45**
Venture	0	1187.5	5.5	0.46	0	0.00	0	0.00	**2035**	**171.37**
Video Pinball	0	17667.9	533936.5	3022.07	572898.27	3242.59	511835	2896.98	925830	5240.18
Wizard of Wor	563.5	4756.5	17862.5	412.57	9157.5	204.96	29059.3	679.60	**64239**	**1519.90**
Yars Revenge	3092.9	54576.9	102557	193.19	84231.14	157.60	166292.3	316.99	**972000**	**1881.96**
Zaxxon	32.5	9173.3	22209.5	242.62	32935.5	359.96	41118	449.47	109140	1193.63
MEAN HNS(%)	0.00	100.00		873.54		956.99		1740.94		7810.1
MEDIAN HNS(%)	0.00	100.00		230.99		191.82		454.91		832.5

Table 8. Score table of SOTA 200M model-free algorithms on HNS(%) (GDI-H^3).

Games	RND	HUMAN	RAINBOW	HNS	IMPALA	HNS	LASER	HNS	GDI-H^3	HNS
Scale			200M		200M		200M		200M	
Alien	227.8	7127.8	9491.7	134.26	15962.1	228.03	35565.9	512.15	**48735**	**703.00**
Amidar	5.8	1719.5	**5131.2**	**299.08**	1554.79	90.39	1829.2	106.4	1065	61.81
Assault	222.4	742	14198.5	2689.78	19148.47	3642.43	21560.4	4106.62	**97155**	**18655.23**
Asterix	210	8503.3	428200	5160.67	300732	3623.67	240090	2892.46	**999999**	**12055.38**
Asteroids	719	47388.7	2712.8	4.27	108590.05	231.14	213025	454.91	**760005**	**1626.94**
Atlantis	12850	29028.1	826660	5030.32	849967.5	5174.39	841200	5120.19	**3837300**	**23639.67**
Bank Heist	14.2	753.1	1358	181.86	1223.15	163.61	569.4	75.14	1380	184.84
Battle Zone	236	37187.5	62010	167.18	20885	55.88	64953.3	175.14	**824360**	**2230.29**
Beam Rider	363.9	16926.5	16850.2	99.54	32463.47	193.81	90881.6	546.52	**422890**	**2551.09**
Berzerk	123.7	2630.4	2545.6	96.62	1852.7	68.98	**25579.5**	**1015.51**	14649	579.46
Bowling	23.1	160.7	30	5.01	59.92	26.76	48.3	18.31	**205.2**	**132.34**
Boxing	0.1	12.1	99.6	829.17	99.96	832.17	**100**	**832.5**	**100**	**832.50**
Breakout	1.7	30.5	417.5	1443.75	787.34	2727.92	747.9	2590.97	**864**	**2994.10**
Centipede	2090.9	12017	8167.3	61.22	11049.75	90.26	**292792**	**2928.65**	195630	1949.80
Chopper Command	811	7387.8	16654	240.89	28255	417.29	761699	11569.27	**999999**	**15192.62**
Crazy Climber	10780.5	36829.4	168788.5	630.80	136950	503.69	167820	626.93	**241170**	**919.76**
Defender	2874.5	18688.9	55105	330.27	185203	1152.93	336953	2112.50	**970540**	**6118.89**
Demon Attack	152.1	1971	111185	6104.40	132826.98	7294.24	133530	7332.89	**787985**	**43313.70**
Double Dunk	-18.6	-16.4	-0.3	831.82	-0.33	830.45	14	1481.82	**24**	**1936.36**
Enduro	0	860.5	2125.9	247.05	0	0.00	0	0.00	14300	1661.82
Fishing Derby	-91.7	-38.8	31.3	232.51	44.85	258.13	45.2	258.79	**65**	**296.22**
Freeway	0	29.6	**34**	**114.86**	0	0.00	0	0.00	**34**	**114.86**
Frostbite	65.2	4334.7	9590.5	223.10	317.75	5.92	5083.5	117.54	**11330**	**263.84**
Gopher	257.6	2412.5	70354.6	3252.91	66782.3	3087.14	114820.7	5316.40	473560	21964.01
Gravitar	173	3351.4	1419.3	39.21	359.5	5.87	1106.2	29.36	**5915**	**180.66**
Hero	1027	30826.4	**55887.4**	**184.10**	33730.55	109.75	31628.7	102.69	38225	124.83
Ice Hockey	-11.2	0.9	1.1	101.65	3.48	121.32	17.4	236.36	**481.90**	
Jamesbond	29	302.8	19809	72.24	601.5	209.09	37999.8	13868.08	**620780**	**226716.95**
Kangaroo	52	3035	**14637.5**	**488.05**	1632	52.97	14308	477.91	14636	488.00
Krull	1598	2665.5	8741.5	669.18	8147.4	613.53	9387.5	729.70	**594540**	**55544.92**
Kung Fu Master	258.5	22736.3	52181	230.99	43375.5	191.82	607443	2701.26	**1666665**	**7413.57**
Montezuma Revenge	0	**4753.3**	384	8.08	0	0.00	0.3	0.01	2500	52.60
Ms Pacman	307.3	6951.6	5380.4	76.35	7342.32	105.88	6565.5	94.19	**11573**	**169.55**
Name This Game	2292.3	8049	13136	188.37	21537.2	334.30	26219.5	415.64	**36296**	**590.68**
Phoenix	761.5	7242.6	108529	1662.80	210996.45	3243.82	519304	8000.84	**959580**	**14794.07**
Pitfall	-229.4	**6463.7**	0	3.43	-1.66	3.40	-0.6	3.42	-4.345	3.36
Pong	-20.7	14.6	20.9	117.85	20.98	118.07	**21**	**118.13**	**21**	**118.13**
Private Eye	24.9	**69571.3**	4234	6.05	98.5	0.11	96.3	0.10	15100	21.68
Qbert	163.9	13455.0	33817.5	253.20	**351200.12**	**2641.14**	21449.6	160.15	28657	214.38
Riverraid	1338.5	17118.0	22920.8	136.77	29608.05	179.15	**40362.7**	**247.31**	28349	171.17
Road Runner	11.5	7845	62041	791.85	57121	729.04	45289	578.00	**999999**	**12765.53**
Robotank	2.2	11.9	61.4	610.31	12.96	110.93	62.1	617.53	**113.4**	**1146.39**
Seaquest	68.4	42054.7	15898.9	37.70	1753.2	4.01	2890.3	6.72	**1000000**	**2381.57**
Skiing	-17098	**-4336.9**	-12957.8	32.44	-10180.38	54.21	-29968.4	-100.86	-6025	86.77
Solaris	1236.3	**12326.7**	3560.3	20.96	2365	10.18	2273.5	9.35	9105	70.95
Space Invaders	148	1668.7	18789	1225.82	43595.78	2857.09	51037.4	3346.45	**154380**	**10142.17**
Star Gunner	664	10250	127029	1318.22	200625	2085.97	321528	3347.21	**677590**	**7061.61**
Surround	-10	6.5	**9.7**	**119.39**	7.56	106.42	8.4	111.52	2.606	76.40
Tennis	-23.8	-8.3	0	153.55	0.55	157.10	12.2	232.26	**24**	**308.39**
Time Pilot	3568	5229.2	12926	563.36	48481.5	2703.84	105316	6125.34	**450810**	**26924.45**
Tutankham	11.4	167.6	241	146.99	292.11	179.71	278.9	171.25	418.2	260.44
Up N Down	533.4	11693.2	125755	1122.08	332546.75	2975.08	345727	3093.19	966590	8656.58
Venture	0	1187.5	5.5	0.46	0	0.00	0	0.00	2000	168.42
Video Pinball	0	17667.9	533936.5	3022.07	572898.27	3242.59	511835	2896.98	**978190**	**5536.54**
Wizard of Wor	563.5	4756.5	17862.5	412.57	9157.5	204.96	29059.3	679.60	63735	1506.59
Yars Revenge	3092.9	54576.9	102557	193.19	84231.14	157.60	166292.3	316.99	968090	1874.36
Zaxxon	32.5	9173.3	22209.5	242.62	32935.5	359.96	41118	449.47	**216020**	**2362.89**
MEAN HNS(%)	0.00	100.00		873.54		956.99		1740.94		**9620.33**
MEDIAN HNS(%)	0.00	100.00		230.99		191.82		454.91		**1146.39**

Table 9. Score table of 10B+ SOTA model-free algorithms on HNS(%).

Games	R2D2	HNS	NGU	HNS	AGENT57	HNS	GDI-I^3	HNS	GDI-H^3	HNS
Scale	10B		35B		100B		200M		200M	
Alien	109038.4	1576.97	248100	3592.35	**297638.17**	**4310.30**	43384	625.45	48735	703.00
Amidar	27751.24	1619.04	17800	1038.35	**29660.08**	**1730.42**	1442	83.81	1065	61.81
Assault	90526.44	17379.53	34800	6654.66	67212.67	12892.66	63876	12250.50	**97155**	**18655.23**
Asterix	999080	12044.30	950700	11460.94	991384.42	11951.51	759910	9160.41	**999999**	**12055.38**
Asteroids	265861.2	568.12	230500	492.36	150854.61	321.70	751970	1609.72	**760005**	**1626.94**
Atlantis	1576068	9662.56	1653600	10141.80	1528841.76	9370.64	3803000	23427.66	**3837300**	**23639.67**
Bank Heist	**46285.6**	**6262.20**	17400	2352.93	23071.5	3120.49	1401	187.68	1380	184.84
Battle Zone	513360	1388.64	691700	1871.27	**934134.88**	**2527.36**	478830	1295.20	824360	2230.29
Beam Rider	128236.08	772.05	63600	381.80	300509.8	1812.19	162100	976.51	**422390**	**2548.07**
Berzerk	34134.8	1356.81	36200	1439.19	**61507.83**	**2448.80**	7607	298.53	14649	579.46
Bowling	196.36	125.92	211.9	137.21	**251.18**	**165.76**	201.9	129.94	205.2	132.34
Boxing	99.16	825.50	99.7	830.00	**100**	**832.50**	**100**	**832.50**	**100**	**832.50**
Breakout	795.36	2755.76	559.2	1935.76	790.4	2738.54	**864**	**2994.10**	**864**	**2994.10**
Centipede	532921.84	5347.83	**577800**	**5799.95**	412847.86	4138.15	155830	1548.84	195630	1949.80
Chopper Command	960648	14594.29	999900	15191.11	999900	15191.11	**999999**	**15192.62**	**999999**	**15192.62**
Crazy Climber	312768	1205.59	313400	1208.11	**565909.85**	**2216.18**	201000	759.39	241170	919.76
Defender	562106	3536.22	664100	4181.16	677642.78	4266.80	893110	5629.27	**970540**	**6118.89**
Demon Attack	143664.6	7890.07	143500	7881.02	143161.44	7862.41	675530	37131.12	**787985**	**43313.70**
Double Dunk	23.12	1896.36	-14.1	204.55	23.93	1933.18	**24**	**1936.36**	**24**	**1936.36**
Enduro	2376.68	276.20	2000	232.42	2367.71	275.16	**14330**	**1665.31**	14300	1661.82
Fishing Derby	81.96	328.28	32	233.84	**86.97**	**337.75**	59	285.71	65	296.22
Freeway	**34**	**114.86**	28.5	96.28	32.59	110.10	**34**	**114.86**	**34**	**114.86**
Frostbite	11238.4	261.70	206400	4832.76	**541280.88**	**12676.32**	10485	244.05	11330	263.84
Gopher	122196	5658.66	113400	5250.47	117777.08	5453.59	**488830**	**22672.63**	473560	21964.01
Gravitar	6750	206.93	14200	441/32	**19213.96**	**599.07**	5905	180.34	5915	180.66
Hero	37030.4	120.82	69400	229.44	**114736.26**	**381.58**	38330	125.18	38225	124.83
Ice Hockey	**71.56**	**683.97**	-4.1	58.68	63.64	618.51	44.94	463.97	47.11	481.90
Jamesbond	23266	8486.85	26600	9704.53	135784.96	49582.16	594500	217118.70	**620780**	**226716.95**
Kangaroo	14112	471.34	**35100**	**1174.92**	24034.16	803.96	14500	484.34	14636	488.90
Krull	145284.8	13460.12	127400	11784.73	251997.31	23456.61	97575	8990.82	**594540**	**55544.92**
Kung Fu Master	200176	889.40	212100	942.45	206845.82	919.07	140440	623.64	**1666665**	**7413.57**
Montezuma Revenge	2504	52.68	**10400**	**218.80**	9352.01	196.75	3000	63.11	2500	52.60
Ms Pacman	29928.2	445.81	40800	609.44	**63994.44**	**958.52**	11536	169.00	11573	169.55
Name This Game	45214.8	745.61	23900	375.35	**54386.77**	**904.94**	34434	558.34	36296	590.68
Phoenix	811621.6	125.11	959100	14786.66	908264.15	14002.29	894460	13789.30	**959580**	**14794.07**
Pitfall	0	3.43	7800	119.97	**18756.01**	**283.66**	0	3.43	-4.3	3.36
Pong	**21**	**118.13**	19.6	114.16	20.67	117.20	**21**	**118.13**	**21**	**118.13**
Private Eye	300	0.40	**100000**	**143.75**	79716.46	114.59	15100	21.68	15100	21.68
Qbert	161000	1210.10	451900	3398.79	**580328.14**	**4365.06**	27800	207.93	28657	214.38
Riverraid	34076.4	207.47	36700	224.10	**63318.67**	**392.79**	28075	169.44	28349	171.17
Road Runner	498660	6365.59	128600	1641.52	243025.8	3102.24	878600	11215.78	**999999**	**12765.53**
Robotank	**132.4**	**1342.27**	9.1	71.13	127.32	1289.90	108.2	1092.78	113.4	1146.39
Seaquest	999991.84	2381.55	**1000000**	**2381.57**	999997.63	2381.56	943910	2247.98	**1000000**	**2381.57**
Skiing	-29970.32	-100.87	-22977.9	-46.08	**-4202.6**	**101.05**	-6774	80.90	-6025	86.77
Solaris	4198.4	26.71	4700	31.23	**44199.93**	**387.39**	11074	88.70	9105	70.95
Space Invaders	55889	3665.48	43400	2844.22	48680.86	3191.48	140460	9226.80	**154380**	**10142.17**
Star Gunner	521728	5435.68	414600	4318.13	**839573.53**	**8751.40**	465750	4851.72	677590	7061.61
Surround	**9.96**	**120.97**	-9.6	2.42	9.5	118.18	-7.8	13.33	2.606	76.40
Tennis	**24**	**308.39**	10.2	219.35	23.84	307.35	**24**	**308.39**	**24**	**308.39**
Time Pilot	348932	20791.28	344700	20536.51	405425.31	24192.24	216770	12834.99	**450810**	**26924.45**
Tutankham	393.64	244.71	191.1	115.04	**2354.91**	**1500.33**	423.9	264.08	418.2	260.44
Up N Down	542918.8	4860.17	620100	5551.77	623805.73	5584.98	**986440**	**8834.45**	966590	8656.58
Venture	1992	167.75	1700	143.16	**2623.71**	**220.94**	2035	171.37	2000	168.42
Video Pinball	483569.72	2737.00	965300	5463.58	**992340.74**	**5616.63**	925830	5240.18	978190	5536.54
Wizard of Wor	133264	3164.81	106200	2519.35	**157306.41**	**3738.20**	64293	1519.90	63735	1506.59
Yars Revenge	918854.32	1778.73	986000	1909.15	**998532.37**	**1933.49**	972000	1881.96	968090	1874.36
Zaxxon	181372	1983.85	111100	1215.07	**249808.9**	**2732.54**	109140	1193.63	216020	2362.89
MEAN HNS(%)		3373.48		3169.07		4762.17		7810.1		**9620.33**
MEDIAN HNS(%)		1342.27		1174.92		**1933.49**		832.5		1146.39

Table 10. Score table of SOTA model-based algorithms on HNS(%). SimPLe (Kaiser et al., 2019) and DreamerV2(Hafner et al., 2020) haven't evaluated all 57 Atari Games in their paper. For fairness, we set the score on those games as N/A, which will not be considered when calculating the median and mean HNS.

Games	MuZero	HNS	DreamerV2	HNS	SimPLe	HNS	GDI-I^3	HNS	GDI-H^3	HNS
Scale	20B		200M		1M		200M		200M	
Alien	**741812.63**	**10747.61**	3483	47.18	616.9	5.64	43384	625.45	48735	703.00
Amidar	**28634.39**	**1670.57**	2028	118.00	74.3	4.00	1442	83.81	1065	61.81
Assault	**143972.03**	**27665.44**	7679	1435.07	527.2	58.66	63876	12250.50	97155	18655.23
Asterix	998425	12036.40	25669	306.98	1128.3	11.07	759910	9160.41	**999999**	**12055.38**
Asteroids	678558.64	1452.42	3064	5.02	793.6	0.16	751970	1609.72	**760005**	**1626.94**
Atlantis	1674767.2	10272.64	989207	6035.05	20992.5	50.33	3803000	23427.66	**3837300**	**23639.67**
Bank Heist	1278.98	171.17	1043	139.23	34.2	2.71	**1401**	**187.68**	1380	184.84
Battle Zone	**848623**	**2295.95**	31225	83.86	4031.2	10.27	478830	1295.20	824360	2230.29
Beam Rider	**454993.53**	**2744.92**	12413	72.75	621.6	1.56	162100	976.51	422390	2548.07
Berzerk	**85932.6**	**3423.18**	751	25.02	N/A	N/A	7607	298.53	14649	579.46
Bowling	**260.13**	**172.26**	48	18.10	30	5.01	202	129.94	205.2	132.34
Boxing	**100**	**832.50**	87	724.17	7.8	64.17	**100**	**832.50**	**100**	**832.50**
Breakout	**864**	**2994.10**	350	1209.38	16.4	51.04	**864**	**2994.10**	**864**	**2994.10**
Centipede	**1159049.27**	**11655.72**	6601	45.44	N/A	N/A	155830	1548.84	195630	1949.80
Chopper Command	991039.7	15056.39	2833	30.74	979.4	2.56	**999999**	**15192.62**	**999999**	**15192.62**
Crazy Climber	**458315.4**	**1786.64**	141424	521.55	62583.6	206.81	201000	759.39	241170	919.76
Defender	839642.95	5291.18	N/A	N/A	N/A	N/A	893110	5629.27	**970540**	**6118.89**
Demon Attack	143964.26	7906.55	2775	144.20	208.1	3.08	675530	37131.12	**787985**	**43313.70**
Double Dunk	23.94	1933.64	22	1845.45	N/A	N/A	**24**	**1936.36**	**24**	**1936.36**
Enduro	2382.44	276.87	2112	245.44	N/A	N/A	**14330**	**1665.31**	14300	1661.82
Fishing Derby	**91.16**	**345.67**	60	286.77	-90.7	1.89	59	285.71	65	296.22
Freeway	33.03	111.59	**34**	**114.86**	16.7	56.42	**34**	**114.86**	**34**	**114.86**
Frostbite	**631378.53**	**14786.59**	15622	364.37	236.9	4.02	10485	244.05	11330	263.84
Gopher	130345.58	6036.85	53853	2487.14	596.8	15.74	**488830**	**22672.6**	473560	21964.01
Gravitar	**6682.7**	**204.81**	3554	106.37	173.4	0.01	5905	180.34	5915	180.66
Hero	**49244.11**	**161.81**	30287	98.19	2656.6	5.47	38330	125.18	38225	124.83
Ice Hockey	**67.04**	**646.61**	29	332.23	-11.6	-3.31	44.94	463.97	47.11	481.90
Jamesbond	41063.25	14986.94	9269	3374.73	100.5	26.11	594500	217118.70	**620780**	**226716.95**
Kangaroo	**16763.6**	**560.23**	11819	394.47	51.2	-0.03	14500	484.34	14636	488.90
Krull	269358.27	25082.93	9687	757.75	2204.8	56.84	97575	8990.82	**594540**	**55544.92**
Kung Fu Master	204824	910.08	66410	294.30	14862.5	64.97	140440	623.64	**1666665**	**7413.57**
Montezuma Revenge	0	0.00	1932	40.65	N/A	N/A	**3000**	**63.11**	2500	52.60
Ms Pacman	**243401.1**	**3658.68**	5651	80.43	1480	17.65	11536	169.00	11573	169.55
Name This Game	**157177.85**	**2690.53**	14472	211.57	2420.7	2.23	34434	558.34	36296	590.68
Phoenix	955137.84	14725.53	13342	194.11	N/A	N/A	894460	13789.30	**959580**	**14794.07**
Pitfall	**0**	**3.43**	-1	3.41	N/A	N/A	**0**	**3.43**	-4.3	3.36
Pong	**21**	**118.13**	19	112.46	12.8	94.90	**21**	**118.13**	**21**	**118.13**
Private Eye	**15299.98**	**21.96**	158	0.19	35	0.01	15100	21.68	15100	21.68
Qbert	**72276**	**542.56**	162023	1217.80	1288.8	8.46	27800	207.93	28657	214.38
Riverraid	**323417.18**	**2041.12**	16249	94.49	1957.8	3.92	28075	169.44	28349	171.17
Road Runner	613411.8	7830.48	88772	1133.09	5640.6	71.86	878600	11215.78	**999999**	**12765.53**
Robotank	**131.13**	**1329.18**	65	647.42	N/A	N/A	108	1092.78	113.4	1146.39
Seaquest	999976.52	2381.51	45898	109.15	683.3	1.46	943910	2247.98	**1000000**	**2381.57**
Skiing	-29968.36	-100.86	-8187	69.83	N/A	N/A	-6774	80.90	**-6025**	**86.77**
Solaris	56.62	-10.64	883	-3.19	N/A	N/A	**11074**	**88.70**	9105	70.95
Space Invaders	74335.3	4878.50	2611	161.96	N/A	N/A	140460	9226.80	**154380**	**10142.17**
Star Gunner	549271.7	5723.01	29219	297.88	N/A	N/A	465750	4851.72	**677590**	**7061.61**
Surround	**9.99**	**121.15**	N/A	N/A	N/A	N/A	-7.8	13.33	2.606	76.40
Tennis	0	153.55	23	301.94	N/A	N/A	**24**	**308.39**	**24**	**308.39**
Time Pilot	**476763.9**	**28486.90**	32404	1735.96	N/A	N/A	216770	12834.99	450810	26924.45
Tutankham	**491.48**	**307.35**	238	145.07	N/A	N/A	424	264.08	418.2	260.44
Up N Down	715545.61	6407.03	648363	5805.03	3350.3	25.24	**986440**	**8834.45**	966590	8656.58
Venture	0.4	0.03	0	0.00	N/A	N/A	**2035**	**171.37**	2000	168.42
Video Pinball	**981791.88**	**5556.92**	22218	125.75	N/A	N/A	925830	5240.18	978190	5536.54
Wizard of Wor	**197126**	**4687.87**	14531	333.11	N/A	N/A	64439	1523.38	63735	1506.59
Yars Revenge	553311.46	1068.72	20089	33.01	5664.3	4.99	**972000**	**1881.96**	968090	1874.36
Zaxxon	**725853.9**	**7940.46**	18295	199.79	N/A	N/A	109140	1193.63	216020	2362.89
MEAN HNS(%)		4994.97		642.49		25.78		7810.1		**9620.33**
MEDIAN HNS(%)		**2041.12**		178.04		5.55		832.5		1146.39

Table 11. Score table of other SOTA algorithms on HNS(%). Go-Explore (Ecoffet et al., 2019) and Muesli (Hessel et al., 2021).

Games	Muesli	HNS	Go-Explore	HNS	GDI-I^3	HNS	GDI-H^3	HNS
Scale	200M		10B		200M		200M	
Alien	139409	2017.12	**959312**	**13899.77**	43384	625.45	48735	703.00
Amidar	**21653**	**1263.18**	19083	1113.22	1442	83.81	1065	61.81
Assault	36963	7070.94	30773	5879.64	63876	12250.50	**97155**	**18655.23**
Asterix	316210	3810.30	999500	12049.37	759910	9160.41	**999999**	**12055.38**
Asteroids	484609	1036.84	112952	240.48	751970	1609.72	**760005**	**1626.94**
Atlantis	1363427	8348.18	286460	1691.24	3803000	23427.66	**3837300**	**23639.67**
Bank Heist	1213	162.24	**3668**	**494.49**	1401	187.68	1380	184.84
Battle Zone	414107	1120.04	**998800**	**2702.36**	478830	1295.20	824360	2230.29
Beam Rider	288870	1741.91	371723	2242.15	162100	976.51	**422390**	**2548.07**
Berzerk	44478	1769.43	**131417**	**5237.69**	7607	298.53	14649	579.46
Bowling	191	122.02	**247**	**162.72**	202	129.94	205.2	132.34
Boxing	99	824.17	91	757.50	**100**	**832.50**	**100**	**832.50**
Breakout	791	2740.63	774	2681.60	**864**	**2994.10**	**864**	**2994.10**
Centipede	**869751**	**8741.20**	613815	6162.78	155830	1548.84	195630	1949.80
Chopper Command	101289	1527.76	996220	15135.16	**999999**	**15192.62**	**999999**	**15192.62**
Crazy Climber	175322	656.88	235600	897.52	201000	759.39	**241170**	**919.76**
Defender	629482	3962.26	N/A	N/A	893110	5629.27	**970540**	**6118.89**
Demon Attack	129544	7113.74	239895	13180.65	675530	37131.12	**787985**	**43313.70**
Double Dunk	-3	709.09	**24**	**1936.36**	**24**	**1936.36**	**24**	**1936.36**
Enduro	2362	274.49	1031	119.81	**14330**	**1665.31**	14300	1661.82
Fishing Derby	51	269.75	**67**	**300.00**	59	285.71	65	296.22
Freeway	33	111.49	**34**	**114.86**	**34**	**114.86**	**34**	**114.86**
Frostbite	301694	7064.73	**999990**	**23420.19**	10485	244.05	11330	263.84
Gopher	104441	4834.72	134244	6217.75	**488830**	**22672.63**	473560	21964.01
Gravitar	11660	361.41	**13385**	**415.68**	5905	180.34	5915	180.66
Hero	37161	121.26	37783	123.34	**38330**	**125.18**	38225	124.83
Ice Hockey	25	299.17	33	365.29	44.94	463.97	**47.11**	**481.90**
Jamesbond	19319	7045.29	200810	73331.26	594500	217118.70	**620780**	**226716.95**
Kangaroo	14096	470.80	**24300**	**812.87**	14500	484.34	14636	488.90
Krull	34221	3056.02	63149	5765.90	97575	8990.82	**594540**	**55544.92**
Kung Fu Master	134689	598.06	24320	107.05	140440	623.64	**1666665**	**7413.57**
Montezuma Revenge	2359	49.63	**24758**	**520.86**	3000	63.11	2500	52.60
Ms Pacman	65278	977.84	**456123**	**6860.25**	11536	169.00	11573	169.55
Name This Game	105043	1784.89	**212824**	**3657.16**	34434	558.34	36296	590.68
Phoenix	805305	12413.69	19200	284.50	894460	13789.30	**959580**	**14794.07**
Pitfall	0	3.43	**7875**	**121.09**	0	3.43	-4.3	3.36
Pong	20	115.30	**21**	**118.13**	**21**	**118.13**	**21**	**118.13**
Private Eye	10323	14.81	**69976**	**100.58**	15100	21.68	15100	21.68
Qbert	157353	1182.66	**999975**	**7522.41**	27800	207.93	28657	214.38
Riverraid	**47323**	**291.42**	35588	217.05	28075	169.44	28349	171.17
Road Runner	327025	4174.55	999900	12764.26	878600	11215.78	**999999**	**12765.53**
Robotank	59	585.57	**143**	**1451.55**	108	1092.78	113.4	1146.39
Seaquest	815970	1943.26	539456	1284.68	943910	2247.98	**1000000**	**2381.57**
Skiing	-18407	-10.26	**-4185**	**101.19**	-6774	80.90	-6025	86.77
Solaris	3031	16.18	**20306**	**171.95**	11074	88.70	9105	70.95
Space Invaders	59602	3909.65	93147	6115.54	140460	9226.80	**154380**	**10142.17**
Star Gunner	214383	2229.49	609580	6352.14	465750	4851.72	**677590**	**7061.61**
Surround	**9**	**115.15**	N/A	N/A	-8	13.33	2.606	76.40
Tennis	12	230.97	**24**	**308.39**	**24**	**308.39**	**24**	**308.39**
Time Pilot	**359105**	**21403.71**	183620	10839.32	216770	12834.99	450810	26924.45
Tutankham	252	154.03	**528**	**330.73**	424	264.08	418.2	260.44
Up N Down	649190	5812.44	553718	4956.94	**986440**	**8834.45**	966590	8656.58
Venture	2104	177.18	**3074**	**258.86**	2035	171.37	2000	168.42
Video Pinball	685436	3879.56	**999999**	**5659.98**	925830	5240.18	978190	5536.54
Wizard of Wor	93291	2211.48	**199900**	**4754.03**	64293	1519.90	63735	1506.59
Yars Revenge	557818	1077.47	**999998**	**1936.34**	972000	1881.96	968090	1874.36
Zaxxon	65325	714.30	18340	200.28	109140	1193.63	**216020**	**2362.89**
MEAN HNS(%)		2538.12		4989.31		7810.1		**9620.33**
MEDIAN HNS(%)		1077.47		**1451.55**		832.5		1146.39

K.6. Atari Games Table of Scores Based on Human World Records

In this part, we detail the raw score of several representative SOTA algorithms , including the SOTA 200M model-free algorithms, SOTA 10B+ model-free algorithms, SOTA model-based algorithms and other SOTA algorithms.[2] Additionally, we calculate the human world records normalized world score (HWRNS) of each game with each algorithm. First of all, we demonstrate the sources of the scores that we used. Random scores are from (Badia et al., 2020a). Human world records (HWR) are from (Hafner et al., 2020; Toromanoff et al., 2019). Rainbow's scores are from (Hessel et al., 2017). IMPALA's scores are from (Espeholt et al., 2018). LASER's scores are from (Schmitt et al., 2020), with no sweep at 200M. As there are many versions of R2D2 and NGU, we use original papers'. R2D2's scores are from (Kapturowski et al., 2018). NGU's scores are from (Badia et al., 2020b). Agent57's scores are from (Badia et al., 2020a). MuZero's scores are from (Schrittwieser et al., 2020). DreamerV2's scores are from (Hafner et al., 2020). SimPLe's scores are from (Kaiser et al., 2019). Go-Explore's scores are from (Ecoffet et al., 2019). Muesli's scores are from (Hessel et al., 2021). In the following, we detail the raw scores and HWRNS of each algorithm on 57 Atari games.

[2] 200M and 10B+ represent the training scale.

Table 12. Score table of SOTA 200M model-free algorithms on HWRNS(%) (GDI-I^3).

Games	RND	HWR	RAINBOW	HWRNS	IMPALA	HWRNS	LASER	HWRNS	GDI-I^3	HWRNS
Scale			200M		200M		200M		200M	
Alien	227.8	**251916**	9491.7	3.68	15962.1	6.25	976.51	14.04	43384	17.15
Amidar	5.8	**104159**	5131.2	4.92	1554.79	1.49	1829.2	1.75	1442	1.38
Assault	222.4	8647	14198.5	165.90	19148.47	224.65	21560.4	253.28	63876	755.57
Asterix	210	**1000000**	428200	42.81	300732	30.06	240090	23.99	759910	75.99
Asteroids	719	**10506650**	2712.8	0.02	108590.05	1.03	213025	2.02	751970	7.15
Atlantis	12850	**10604840**	826660	7.68	849967.5	7.90	841200	7.82	3803000	35.78
Bank Heist	14.2	**82058**	1358	1.64	1223.15	1.47	569.4	0.68	1401	1.69
Battle Zone	236	801000	62010	7.71	20885	2.58	64953.3	8.08	478830	59.77
Beam Rider	363.9	**999999**	16850.2	1.65	32463.47	3.21	90881.6	9.06	162100	16.18
Berzerk	123.7	**1057940**	2545.6	0.23	1852.7	0.16	25579.5	2.41	7607	0.71
Bowling	23.1	**300**	30	2.49	59.92	13.30	48.3	9.10	201.9	64.57
Boxing	0.1	**100**	99.6	99.60	99.96	99.96	**100**	**100.00**	**100**	**100.00**
Breakout	1.7	**864**	417.5	48.22	787.34	91.11	747.9	86.54	**864**	**100.00**
Centipede	2090.9	**1301709**	8167.3	0.47	11049.75	0.69	292792	22.37	155830	11.83
Chopper Command	811	**999999**	16654	1.59	28255	2.75	761699	76.15	**999999**	**100.00**
Crazy Climber	10780.5	219900	168788.5	75.56	136950	60.33	167820	75.10	201000	90.96
Defender	2874.5	**6010500**	55105	0.87	185203	3.03	336953	5.56	893110	14.82
Demon Attack	152.1	**1556345**	111185	7.13	132826.98	8.53	133530	8.57	675530	43.40
Double Dunk	-18.6	21	-0.3	46.21	-0.33	46.14	14	82.32	**24**	**107.58**
Enduro	0	9500	2125.9	22.38	0	0.00	0	0.00	**14330**	**150.84**
Fishing Derby	-91.7	**71**	31.3	75.60	44.85	83.93	45.2	84.14	59	92.89
Freeway	0	**38**	34	89.47	0	0.00	0	0.00	34	89.47
Frostbite	65.2	**454830**	9590.5	2.09	317.75	0.06	5083.5	1.10	10485	2.29
Gopher	257.6	355040	70354.6	19.76	66782.3	18.75	114820.7	32.29	**488830**	**137.71**
Gravitar	173	**162850**	1419.3	0.77	359.5	0.11	1106.2	0.57	5905	3.52
Hero	1027	**1000000**	55887.4	5.49	33730.55	3.27	31628.7	3.06	38330	3.73
Ice Hockey	-11.2	36	1.1	26.06	3.48	31.10	17.4	60.59	44.92	118.94
Jamesbond	29	45550	19809	43.45	601.5	1.26	37999.8	83.41	594500	1305.93
Kangaroo	52	**1424600**	14637.5	1.02	1632	0.11	14308	1.00	14500	1.01
Krull	1598	104100	8741.5	6.97	8147.4	6.39	9387.5	7.60	97575	93.63
Kung Fu Master	258.5	1000000	52181	5.19	43375.5	4.31	607443	60.73	140440	14.02
Montezuma Revenge	0	**1219200**	384	0.03	0	0.00	0.3	0.00	3000	0.25
Ms Pacman	307.3	**290090**	5380.4	1.75	7342.32	2.43	6565.5	2.16	11536	3.87
Name This Game	2292.3	25220	13136	47.30	21537.2	83.94	26219.5	104.36	34434	140.19
Phoenix	761.5	**4014440**	108529	2.69	210996.45	5.24	519304	12.92	894460	22.27
Pitfall	-229.4	**114000**	0	0.20	-1.66	0.20	-0.6	0.20	**0**	0.20
Pong	-20.7	**21**	20.9	99.76	20.98	99.95	**21**	**100.00**	**21**	**100.00**
Private Eye	24.9	**101800**	4234	4.14	98.5	0.07	96.3	0.07	15100	14.81
Qbert	163.9	**2400000**	33817.5	1.40	351200.12	14.63	21449.6	0.89	27800	1.15
Riverraid	1338.5	**1000000**	22920.8	2.16	29608.05	2.83	40362.7	3.91	28075	2.68
Road Runner	11.5	**2038100**	62041	3.04	57121	2.80	45289	2.22	878600	43.11
Robotank	2.2	76	61.4	80.22	12.96	14.58	62.1	81.17	108.2	143.63
Seaquest	68.4	999999	15898.9	1.58	1753.2	0.17	2890.3	0.28	943910	94.39
Skiing	-17098	**-3272**	-12957.8	29.95	-10180.38	50.03	-29968.4	-93.09	-6774	74.67
Solaris	1236.3	**111420**	3560.3	2.11	2365	1.02	2273.5	0.94	11074	8.93
Space Invaders	148	**621535**	18789	3.00	43595.78	6.99	51037.4	8.19	140460	22.58
Star Gunner	664	77400	127029	164.67	200625	260.58	321528	418.14	465750	606.09
Surround	-10	9.6	**9.7**	**100.51**	7.56	89.59	8.4	93.88	-7.8	11.22
Tennis	-23.8	21	0	53.13	0.55	54.35	12.2	80.36	**24**	**106.70**
Time Pilot	3568	65300	12926	15.16	48481.5	72.76	105316	164.82	216770	345.37
Tutankham	11.4	**5384**	241	4.27	292.11	5.22	278.9	4.98	423.9	7.68
Up N Down	533.4	82840	125755	152.14	332546.75	403.39	345727	419.40	**986440**	**1197.85**
Venture	0	**38900**	5.5	0.01	0	0.00	0	0.00	2000	5.23
Video Pinball	0	**89218328**	533936.5	0.60	572898.27	0.64	511835	0.57	925830	1.04
Wizard of Wor	563.5	**395300**	17862.5	4.38	9157.5	2.18	29059.3	7.22	64439	16.14
Yars Revenge	3092.9	**15000105**	102557	0.66	84231.14	0.54	166292.3	1.09	972000	6.46
Zaxxon	32.5	83700	22209.5	26.51	32935.5	39.33	41118	49.11	109140	130.41
MEAN HWRNS(%)	0.00	100.00		28.39		34.52		45.39		117.98
MEDIAN HWRNS(%)	0.00	**100.00**		4.92		4.31		8.08		35.78

Table 13. Score table of SOTA 200M model-free algorithms on HWRNS(%) (GDI-H^3).

Games	RND	HWR	RAINBOW	HWRNS	IMPALA	HWRNS	LASER	HWRNS	GDI-H^3	HWRNS
Scale			200M		200M		200M		200M	
Alien	227.8	**251916**	9491.7	3.68	15962.1	6.25	976.51	14.04	48735	19.27
Amidar	5.8	**104159**	5131.2	4.92	1554.79	1.49	1829.2	1.75	1065	1.02
Assault	222.4	8647	14198.5	165.90	19148.47	224.65	21560.4	253.28	**97155**	**1150.59**
Asterix	210	**1000000**	428200	42.81	300732	30.06	240090	23.99	999999	100.00
Asteroids	719	**10506650**	2712.8	0.02	108590.05	1.03	213025	2.02	760005	7.23
Atlantis	12850	**10604840**	826660	7.68	849967.5	7.90	841200	7.82	3837300	36.11
Bank Heist	14.2	**82058**	1358	1.64	1223.15	1.47	569.4	0.68	1380	1.66
Battle Zone	236	801000	62010	7.71	20885	2.58	64953.3	8.08	**824360**	102.92
Beam Rider	363.9	**999999**	16850.2	1.65	32463.47	3.21	90881.6	9.06	422390	42.22
Berzerk	123.7	**1057940**	2545.6	0.23	1852.7	0.16	25579.5	2.41	14649	1.37
Bowling	23.1	**300**	30	2.49	59.92	13.30	48.3	9.10	205.2	65.76
Boxing	0.1	**100**	99.6	99.60	99.96	99.96	**100**	**100.00**	**100**	**100.00**
Breakout	1.7	**864**	417.5	48.22	787.34	91.11	747.9	86.54	**864**	**100.00**
Centipede	2090.9	**1301709**	8167.3	0.47	11049.75	0.69	292792	22.37	195630	14.89
Chopper Command	811	**999999**	16654	1.59	28255	2.75	761699	76.15	**999999**	**100.00**
Crazy Climber	10780.5	219900	168788.5	75.56	136950	60.33	167820	75.10	**241170**	**110.17**
Defender	2874.5	**6010500**	55105	0.87	185203	3.03	336953	5.56	970540	16.11
Demon Attack	152.1	**1556345**	111185	7.13	132826.98	8.53	133530	8.57	**787985**	**50.63**
Double Dunk	-18.6	21	-0.3	46.21	-0.33	46.14	14	82.32	**24**	**107.58**
Enduro	0	9500	2125.9	22.38	0	0.00	0	0.00	14300	150.53
Fishing Derby	-91.7	**71**	31.3	75.60	44.85	83.93	45.2	84.14	65	96.31
Freeway	0	**38**	34	89.47	0	0.00	0	0.00	34	89.47
Frostbite	65.2	**454830**	9590.5	2.09	317.75	0.06	5083.5	1.10	11330	2.48
Gopher	257.6	355040	70354.6	19.76	66782.3	18.75	114820.7	32.29	473560	133.41
Gravitar	173	**162850**	1419.3	0.77	359.5	0.11	1106.2	0.57	5915	3.53
Hero	1027	**1000000**	55887.4	5.49	33730.55	3.27	31628.7	3.06	38225	3.72
Ice Hockey	-11.2	36	1.1	26.06	3.48	31.10	17.4	60.59	**47.11**	**123.54**
Jamesbond	29	45550	19809	43.45	601.5	1.26	37999.8	83.41	**620780**	**1363.66**
Kangaroo	52	**1424600**	14637.5	1.02	1632	0.11	14308	1.00	14636	1.02
Krull	1598	104100	8741.5	6.97	8147.4	6.39	9387.5	7.60	**594540**	**578.47**
Kung Fu Master	258.5	1000000	52181	5.19	43375.5	4.31	607443	60.73	**1666665**	**166.68**
Montezuma Revenge	0	**1219200**	384	0.03	0	0.00	0.3	0.00	2500	0.21
Ms Pacman	307.3	**290090**	5380.4	1.75	7342.32	2.43	6565.5	2.16	11573	3.89
Name This Game	2292.3	25220	13136	47.30	21537.2	83.94	26219.5	104.36	**36296**	**148.31**
Phoenix	761.5	**4014440**	108529	2.69	210996.45	5.24	519304	12.92	959580	23.89
Pitfall	-229.4	**114000**	0	0.20	-1.66	0.20	-0.6	0.20	-4.3	0.20
Pong	-20.7	**21**	20.9	99.76	20.98	99.95	**21**	**100.00**	**21**	**100.00**
Private Eye	24.9	**101800**	4234	4.14	98.5	0.07	96.3	0.07	15100	14.81
Qbert	163.9	**2400000**	33817.5	1.40	351200.12	14.63	21449.6	0.89	28657	1.19
Riverraid	1338.5	**1000000**	22920.8	2.16	29608.05	2.83	40362.7	3.91	28349	2.70
Road Runner	11.5	**2038100**	62041	3.04	57121	2.80	45289	2.22	999999	49.06
Robotank	2.2	76	61.4	80.22	12.96	14.58	62.1	81.17	**113.4**	**150.68**
Seaquest	68.4	999999	15898.9	1.58	1753.2	0.17	2890.3	0.28	**1000000**	**100.00**
Skiing	-17098	**-3272**	-12957.8	29.95	-10180.38	50.03	-29968.4	-93.09	-6025	86.77
Solaris	1236.3	**111420**	3560.3	2.11	2365	1.02	2273.5	0.94	9105	7.14
Space Invaders	148	**621535**	18789	3.00	43595.78	6.99	51037.4	8.19	154380	24.82
Star Gunner	664	77400	127029	164.67	200625	260.58	321528	418.14	**677590**	**882.15**
Surround	-10	9.6	**9.7**	**100.51**	7.56	89.59	8.4	93.88	2.606	64.32
Tennis	-23.8	21	0	53.13	0.55	54.35	12.2	80.36	**24**	**106.70**
Time Pilot	3568	65300	12926	15.16	48481.5	72.76	105316	164.82	**450810**	**724.49**
Tutankham	11.4	**5384**	241	4.27	292.11	5.22	278.9	4.98	418.2	7.57
Up N Down	533.4	82840	125755	152.14	332546.75	403.39	345727	419.40	966590	1173.73
Venture	0	**38900**	5.5	0.01	0	0.00	0	0.00	2000	5.14
Video Pinball	0	**89218328**	533936.5	0.60	572898.27	0.64	511835	0.57	978190	1.10
Wizard of Wor	563.5	**395300**	17862.5	4.38	9157.5	2.18	29059.3	7.22	63735	16.00
Yars Revenge	3092.9	**15000105**	102557	0.66	84231.14	0.54	166292.3	1.09	968090	6.43
Zaxxon	32.5	83700	22209.5	26.51	32935.5	39.33	41118	49.11	**216020**	**258.15**
MEAN HWRNS(%)	0.00	100.00		28.39		34.52		45.39		**154.27**
MEDIAN HWRNS(%)	0.00	**100.00**		4.92		4.31		8.08		50.63

Table 14. Score table of SOTA 10B+ model-free algorithms on HWRNS(%).

Games	R2D2	HWRNS	NGU	HWRNS	AGENT57	HWRNS	GDI-I^3	HWRNS	GDI-H^3	HWRNS
Scale	10B		35B		100B		200M		200M	
Alien	109038.4	43.23	248100	98.48	**297638.17**	**118.17**	43384	17.15	48735	19.27
Amidar	27751.24	26.64	17800	17.08	**29660.08**	**28.47**	1442	1.38	1065	1.02
Assault	90526.44	1071.91	34800	410.44	67212.67	795.17	63876	755.57	**97155**	**1150.59**
Asterix	999080	99.91	950700	95.07	991384.42	99.14	759910	75.99	**999999**	**100.00**
Asteroids	265861.2	2.52	230500	2.19	150854.61	1.43	751970	7.15	**760005**	**7.23**
Atlantis	1576068	14.76	1653600	15.49	1528841.76	14.31	3803000	35.78	**3837300**	**36.11**
Bank Heist	**46285.6**	**56.40**	17400	21.19	23071.5	28.10	1401	1.69	1380	1.66
Battle Zone	513360	64.08	691700	86.35	**934134.88**	**116.63**	478830	59.77	824360	102.92
Beam Rider	128236.08	12.79	63600	6.33	300509.8	30.03	162100	16.18	**422390**	**42.22**
Berzerk	34134.8	3.22	36200	3.41	**61507.83**	**5.80**	7607	0.71	14649	1.37
Bowling	196.36	62.57	211.9	68.18	**251.18**	**82.37**	201.9	64.57	205.2	65.76
Boxing	99.16	99.16	99.7	99.70	**100**	**100.00**	**100**	**100.00**	**100**	**100.00**
Breakout	795.36	92.04	559.2	64.65	790.4	91.46	**864**	**100.00**	**864**	**100.00**
Centipede	532921.84	40.85	**577800**	**44.30**	412847.86	31.61	155830	11.83	195630	14.89
Chopper Command	960648	96.06	999900	99.99	999900	99.99	**999999**	**100.00**	**999999**	**100.00**
Crazy Climber	312768	144.41	313400	144.71	**565909.85**	**265.46**	201000	90.96	241170	110.17
Defender	562106	9.31	664100	11.01	677642.78	11.23	893110	14.82	**970540**	**16.11**
Demon Attack	143664.6	9.22	143500	9.21	143161.44	9.19	675530	43.40	**787985**	**50.63**
Double Dunk	23.12	105.35	-14.1	11.36	23.93	107.40	**24**	**107.58**	**24**	**107.58**
Enduro	2376.68	25.02	2000	21.05	2367.71	24.92	**14330**	**150.84**	14300	150.53
Fishing Derby	81.96	106.74	32	76.03	**86.97**	**109.82**	59	92.89	65	96.31
Freeway	**34**	**89.47**	28.5	75.00	32.59	85.76	**34**	**89.47**	**34**	**89.47**
Frostbite	11238.4	2.46	206400	45.37	**541280.88**	**119.01**	10485	2.29	11330	2.48
Gopher	122196	34.37	113400	31.89	117777.08	33.12	**488830**	**137.71**	473560	133.41
Gravitar	6750	4.04	14200	8.62	**19213.96**	**11.70**	5905	3.52	5915	3.53
Hero	37030.4	3.60	69400	6.84	**114736.26**	**11.38**	38330	3.73	38225	3.72
Ice Hockey	**71.56**	**175.34**	-4.1	15.04	63.64	158.56	37.89	118.94	47.11	123.54
Jamesbond	23266	51.05	26600	58.37	135784.96	298.23	594500	1305.93	**620780**	**1363.66**
Kangaroo	14112	0.99	**35100**	**2.46**	24034.16	1.68	14500	1.01	14636	1.02
Krull	145284.8	140.18	127400	122.73	251997.31	244.29	97575	93.63	**594540**	**578.47**
Kung Fu Master	200176	20.00	212100	21.19	206845.82	20.66	140440	14.02	**1666665**	**166.68**
Montezuma Revenge	2504	0.21	**10400**	**0.85**	9352.01	0.77	3000	0.25	2500	0.21
Ms Pacman	29928.2	10.22	40800	13.97	**63994.44**	**21.98**	11536	3.87	11573	3.89
Name This Game	45214.8	187.21	23900	94.24	**54386.77**	**227.21**	34434	140.19	36296	148.31
Phoenix	811621.6	20.20	**959100**	**23.88**	908264.15	22.61	894460	22.27	959580	23.89
Pitfall	0	0.20	7800	7.03	**18756.01**	**16.62**	0	0.20	-4.3	0.20
Pong	**21**	**100.00**	19.6	96.64	20.67	99.21	**21**	**100.00**	**21**	**100.00**
Private Eye	300	0.27	**100000**	**98.23**	79716.46	78.30	15100	14.81	15100	14.81
Qbert	161000	6.70	451900	18.82	**580328.14**	**24.18**	27800	1.15	28657	1.19
Riverraid	34076.4	3.28	36700	3.54	**63318.67**	**6.21**	28075	2.68	28349	2.70
Road Runner	498660	24.47	128600	6.31	243025.8	11.92	878600	43.11	**999999**	**49.06**
Robotank	**132.4**	**176.42**	9.1	9.35	127.32	169.54	108	143.63	113.4	150.68
Seaquest	999991.84	100.00	**1000000**	**100.00**	999997.63	100.00	943910	94.39	**1000000**	**100.00**
Skiing	-29970.32	-93.10	-22977.9	-42.53	**-4202.6**	**93.27**	-6774	74.67	-6025	86.77
Solaris	4198.4	2.69	4700	3.14	**44199.93**	**38.99**	11074	8.93	9105	7.14
Space Invaders	55889	8.97	43400	6.96	48680.86	7.81	140460	22.58	**154380**	**24.82**
Star Gunner	521728	679.03	414600	539.43	**839573.53**	**1093.24**	465750	606.09	677590	882.15
Surround	**9.96**	**101.84**	-9.6	2.04	9.5	99.49	-7.8	11.22	2.606	64.32
Tennis	**24**	**106.70**	10.2	75.89	23.84	106.34	**24**	**106.70**	**24**	**106.70**
Time Pilot	348932	559.46	344700	552.60	**405425.31**	**650.97**	216770	345.37	450810	724.49
Tutankham	393.64	7.11	191.1	3.34	**2354.91**	**43.62**	423.9	7.68	418.2	7.57
Up N Down	542918.8	658.98	620100	752.75	623805.73	757.26	**986440**	**1197.85**	966590	1173.73
Venture	1992	5.12	1700	4.37	**2623.71**	**6.74**	2000	5.23	2000	5.14
Video Pinball	483569.72	0.54	965300	1.08	**992340.74**	**1.11**	925830	1.04	978190	1.10
Wizard of Wor	133264	33.62	106200	26.76	**157306.41**	**39.71**	64439	16.14	63735	16.00
Yars Revenge	918854.32	6.11	986000	6.55	**998532.37**	**6.64**	972000	6.46	968090	6.43
Zaxxon	181372	216.74	111100	132.75	**249808.9**	**298.53**	109140	130.41	216020	258.15
MEAN HWRNS(%)		98.78		76.00		125.92		117.98		**154.27**
MEDIAN HWRNS(%)		33.62		21.19		43.62		35.78		**50.63**

Table 15. Score table of SOTA model-based algorithms on HWRNS(%). SimPLe (Kaiser et al., 2019) and DreamerV2(Hafner et al., 2020) haven't evaluated all 57 Atari Games in their paper. For fairness, we set the score on those games as N/A, which will not be considered when calculating the median and mean HWRNS and human world record breakthrough (HWRB).

Games	MuZero	HWRNS	DreamerV2	HWRNS	SimPLe	HWRNS	GDI-I^3	HWRNS	GDI-H^3	HWRNS
Scale	20B		200M		1M		200M		200M	
Alien	**741812.63**	**294.64**	3483	1.29	616.9	0.15	43384	17.15	48735	19.27
Amidar	**28634.39**	**27.49**	2028	1.94	74.3	0.07	1442	1.38	1065	1.02
Assault	**143972.03**	**1706.31**	7679	88.51	527.2	3.62	63876	755.57	97155	1150.59
Asterix	998425	99.84	25669	2.55	1128.3	0.09	759910	75.99	**999999**	**100.00**
Asteroids	678558.64	6.45	3064	0.02	793.6	0.00	751970	7.15	**760005**	**7.23**
Atlantis	1674767.2	15.69	989207	9.22	20992.5	0.08	3803000	35.78	**3837300**	**36.11**
Bank Heist	1278.98	1.54	1043	1.25	34.2	0.02	**1401**	**1.69**	1380	1.66
Battle Zone	**848623**	**105.95**	31225	3.87	4031.2	0.47	478830	59.77	824360	102.92
Beam Rider	**454993.53**	**45.48**	12413	1.21	621.6	0.03	162100	16.18	422390	42.22
Berzerk	**85932.6**	**8.11**	751	0.06	N/A	N/A	7607	0.71	14649	1.37
Bowling	**260.13**	**85.60**	48	8.99	30	2.49	202	64.57	205.2	65.76
Boxing	**100**	**100.00**	87	86.99	7.8	7.71	**100**	**100.00**	**100**	**100.00**
Breakout	**864**	**100.00**	350	40.39	16.4	1.70	**864**	**100.00**	**864**	100.00
Centipede	**1159049.27**	**89.02**	6601	0.35	N/A	N/A	155830	11.83	195630	14.89
Chopper Command	991039.7	99.10	2833	0.20	979.4	0.02	**999999**	**100.00**	**999999**	**100.00**
Crazy Climber	**458315.4**	**214.01**	141424	62.47	62583.6	24.77	201000	90.96	241170	110.17
Defender	839642.95	13.93	N/A	N/A	N/A	N/A	893110	14.82	**970540**	**16.11**
Demon Attack	143964.26	9.24	2775	0.17	208.1	0.00	675530	43.40	**787985**	**50.63**
Double Dunk	23.94	107.42	22	102.53	N/A	N/A	**24**	**107.58**	**24**	**107.58**
Enduro	2382.44	25.08	2112	22.23	N/A	N/A	**14330**	**150.84**	14300	150.53
Fishing Derby	**91.16**	**112.39**	93.24	286.77	-90.7	0.61	59	92.89	65	96.31
Freeway	33.03	86.92	**34**	**89.47**	16.7	43.95	**34**	**89.47**	**34**	**89.47**
Frostbite	**631378.53**	**138.82**	15622	3.42	236.9	0.04	10485	2.29	11330	2.48
Gopher	130345.58	36.67	53853	15.11	596.8	0.10	**488830**	**137.71**	473560	133.41
Gravitar	**6682.7**	**4.00**	3554	2.08	173.4	0.00	5905	3.52	5915	3.53
Hero	**49244.11**	**4.83**	30287	2.93	2656.6	0.16	38330	3.73	38225	3.72
Ice Hockey	**67.04**	**165.76**	29	85.17	-11.6	-0.85	38	118.94	47.11	123.54
Jamesbond	41063.25	90.14	9269	20.30	100.5	0.16	594500	1305.93	**620780**	**1363.66**
Kangaroo	**16763.6**	**1.17**	11819	0.83	51.2	0.00	14500	1.01	14636	1.02
Krull	269358.27	261.22	9687	7.89	2204.8	0.59	97575	93.63	**594540**	**578.47**
Kung Fu Master	204824	20.46	66410	6.62	14862.5	1.46	140440	14.02	**1666665**	**166.68**
Montezuma Revenge	0	0.00	1932	0.16	N/A	N/A	**3000**	**0.25**	2500	0.21
Ms Pacman	**243401.1**	**83.89**	5651	1.84	1480	0.40	11536	3.87	11573	3.89
Name This Game	**157177.85**	**675.54**	14472	53.12	2420.7	0.56	34434	140.19	36296	148.31
Phoenix	**955137.84**	**23.78**	13342	0.31	N/A	N/A	894460	22.27	959580	23.89
Pitfall	**0**	**0.20**	-1	0.20	N/A	N/A	**0**	**0.20**	-4.3	0.20
Pong	**21**	100.00	19	95.20	12.8	80.34	**21**	**100.00**	**21**	**100.00**
Private Eye	**15299.98**	**15.01**	158	0.13	35	0.01	15100	14.81	15100	14.81
Qbert	**72276**	**3.00**	162023	6.74	1288.8	0.05	27800	1.15	28657	1.19
Riverraid	**323417.18**	**32.25**	16249	1.49	1957.8	0.06	28075	2.68	28349	2.70
Road Runner	613411.8	30.10	88772	4.36	5640.6	0.28	878600	43.11	**999999**	**49.06**
Robotank	**131.13**	**174.70**	65	85.09	N/A	N/A	108	143.63	113.4	150.68
Seaquest	999976.52	100.00	45898	4.58	683.3	0.06	943910	94.39	**1000000**	**100.00**
Skiing	-29968.36	-93.09	-8187	64.45	N/A	N/A	-6774	74.67	**-6025**	**86.77**
Solaris	56.62	-1.07	883	-0.32	N/A	N/A	**11074**	**8.93**	9105	7.14
Space Invaders	74335.3	11.94	2611	0.40	N/A	N/A	140460	22.58	**154380**	**24.82**
Star Gunner	549271.7	714.93	29219	37.21	N/A	N/A	465750	606.09	**677590**	**882.15**
Surround	**9.99**	**101.99**	N/A	N/A	N/A	N/A	-8	11.22	2.606	64.32
Tennis	0	53.13	23	104.46	N/A	N/A	**24**	**106.70**	**24**	**106.70**
Time Pilot	**476763.9**	**766.53**	32404	46.71	N/A	N/A	216770	345.37	450810	724.49
Tutankham	**491.48**	**8.94**	238	4.22	N/A	N/A	424	7.68	418.2	7.57
Up N Down	715545.61	868.72	648363	787.09	3350.3	3.42	**986440**	**1197.85**	966590	1173.73
Venture	0.4	0.00	0	0.00	N/A	N/A	**2030**	**5.23**	2000	5.14
Video Pinball	**981791.88**	**1.10**	22218	0.02	N/A	N/A	925830	1.04	978190	1.10
Wizard of Wor	**197126**	**49.80**	14531	3.54	N/A	N/A	64439	16.14	63735	16.00
Yars Revenge	553311.46	3.67	20089	0.11	5664.3	0.02	**972000**	**6.46**	968090	6.43
Zaxxon	**725853.9**	**867.51**	18295	21.83	N/A	N/A	109140	130.41	216020	258.15
MEAN HWRNS(%)		152.10		4.29		4.80		117.98		**154.27**
MEDIAN HWRNS(%)		49.80		4.29		0.13		**35.78**		50.63

Table 16. Score table of other SOTA algorithms on HWRNS(%). Go-Explore (Ecoffet et al., 2019) and Muesli (Hessel et al., 2021).

Games	Muesli	HWRNS	Go-Explore	HWRNS	GDI-I^3	HWRNS	GDI-H^3	HWRNS
Scale		200M	10B		200M		200M	
Alien	139409	55.30	**959312**	**381.06**	43384	17.15	48735	19.27
Amidar	**21653**	**20.78**	19083	18.32	1442	1.38	1065	1.02
Assault	36963	436.11	30773	362.64	63876	755.57	**97155**	**1150.59**
Asterix	316210	31.61	999500	99.95	759910	75.99	**999999**	**100.00**
Asteroids	484609	4.61	112952	1.07	751970	7.15	**760005**	**7.23**
Atlantis	1363427	12.75	286460	2.58	3803000	35.78	**3837300**	**36.11**
Bank Heist	1213	1.46	**3668**	**4.45**	1401	1.69	1380	1.66
Battle Zone	414107	51.68	**998800**	**124.70**	478830	59.77	824360	102.92
Beam Rider	288870	28.86	371723	37.15	162100	16.18	**422390**	**42.22**
Berzerk	44478	4.19	**131417**	**12.41**	7607	0.71	14649	1.37
Bowling	191	60.64	**247**	**80.86**	202	64.57	205.2	65.76
Boxing	99	99.00	91	90.99	**100**	**100.00**	**100**	**100.00**
Breakout	791	91.53	774	89.56	**864**	**100.00**	**864**	**100.00**
Centipede	**869751**	**66.76**	613815	47.07	155830	11.83	195630	14.89
Chopper Command	101289	10.06	996220	99.62	**999999**	**100.00**	**999999**	**100.00**
Crazy Climber	175322	78.68	235600	107.51	201000	90.96	**241170**	**110.17**
Defender	629482	10.43	N/A	N/A	893110	14.82	**970540**	**16.11**
Demon Attack	129544	8.31	239895	15.41	675530	43.40	**787985**	**50.63**
Double Dunk	-3	39.39	**24**	**107.58**	**24**	**107.58**	**24**	**107.58**
Enduro	2362	24.86	1031	10.85	**14330**	**150.84**	14300	150.53
Fishing Derby	51	87.71	**67**	**97.54**	59	92.89	65	96.31
Freeway	33	86.84	**34**	**89.47**	**34**	**89.47**	**34**	**89.47**
Frostbite	301694	66.33	**999990**	**219.88**	10485	2.29	11330	2.48
Gopher	104441	29.37	134244	37.77	**488830**	**137.71**	473560	133.41
Gravitar	11660	7.06	**13385**	**8.12**	5905	3.52	5915	3.53
Hero	37161	3.62	37783	3.68	**38330**	**3.73**	38225	3.72
Ice Hockey	25	76.69	33	93.64	45	118.94	**47.11**	**123.54**
Jamesbond	19319	42.38	200810	441.07	594500	1305.93	**620780**	**1363.66**
Kangaroo	14096	0.99	**24300**	**1.70**	14500	1.01	14636	1.02
Krull	34221	31.83	63149	60.05	97575	93.63	**594540**	**578.47**
Kung Fu Master	134689	13.45	24320	2.41	140440	14.02	**1666665**	**166.68**
Montezuma Revenge	2359	0.19	**24758**	**2.03**	3000	0.25	2500	0.21
Ms Pacman	65278	22.42	**456123**	**157.30**	11536	3.87	11573	3.89
Name This Game	105043	448.15	**212824**	**918.24**	34434	140.19	36296	148.31
Phoenix	805305	20.05	19200	0.46	894460	22.27	**959580**	**23.89**
Pitfall	0	0.20	**7875**	**7.09**	0	0.2	-4.3	0.20
Pong	20	97.60	**21**	**100.00**	**21**	**100**	**21**	**100.00**
Private Eye	10323	10.12	**69976**	**68.73**	15100	14.81	15100	14.81
Qbert	157353	6.55	**999975**	**41.66**	27800	1.15	28657	1.19
Riverraid	**47323**	**4.60**	35588	3.43	28075	2.68	28349	2.70
Road Runner	327025	16.05	999900	49.06	878600	43.11	**999999**	**49.06**
Robotank	59	76.96	**143**	**190.79**	108	143.63	113.4	150.68
Seaquest	815970	81.60	539456	53.94	943910	94.39	**1000000**	**100.00**
Skiing	-18407	-9.47	**-4185**	**93.40**	-6774	74.67	-6025	86.77
Solaris	3031	1.63	**20306**	**17.31**	11074	8.93	9105	7.14
Space Invaders	59602	9.57	93147	14.97	140460	22.58	**154380**	**24.82**
Star Gunner	214383	278.51	609580	793.52	465750	606.09	**677590**	**882.15**
Surround	**9**	**96.94**	N/A	N/A	-8	11.22	2.606	64.32
Tennis	12	79.91	**24**	**106.7**	**24**	**106.70**	**24**	**106.70**
Time Pilot	359105	575.94	183620	291.67	216770	345.37	**450810**	**724.49**
Tutankham	252	4.48	**528**	**9.62**	424	7.68	418.2	7.57
Up N Down	649190	788.10	553718	672.10	**986440**	**1197.85**	966590	1173.73
Venture	2104	5.41	**3074**	**7.90**	2035	5.23	2000	5.14
Video Pinball	685436	0.77	**999999**	**1.12**	925830	1.04	978190	1.10
Wizard of Wor	93291	23.49	**199900**	**50.50**	64293	16.14	63735	16.00
Yars Revenge	557818	3.70	**999998**	**6.65**	972000	6.46	968090	6.43
Zaxxon	65325	78.04	18340	21.88	109140	130.41	**216020**	**258.15**
MEAN HWRNS(%)		75.52		116.89		117.98		**154.27**
MEDIAN HWRNS(%)		24.86		50.50		35.78		**50.63**

K.7. Atari Games Table of Scores Based on SABER

In this part, we detail the raw score of several representative SOTA algorithms , including the SOTA 200M model-free algorithms, SOTA 10B+ model-free algorithms, SOTA model-based algorithms and other SOTA algorithms.[3] Additionally, we calculate the capped human world records normalized world score (CHWRNS) or called SABER (Toromanoff et al., 2019) of each game with each algorithm. First of all, we demonstrate the sources of the scores that we used. Random scores are from (Badia et al., 2020a). Human world records (HWR) are from (Hafner et al., 2020; Toromanoff et al., 2019). Rainbow's scores are from (Hessel et al., 2017). IMPALA's scores are from (Espeholt et al., 2018). LASER's scores are from (Schmitt et al., 2020), with no sweep at 200M. As there are many versions of R2D2 and NGU, we use original papers'. R2D2's scores are from (Kapturowski et al., 2018). NGU's scores are from (Badia et al., 2020b). Agent57's scores are from (Badia et al., 2020a). MuZero's scores are from (Schrittwieser et al., 2020). DreamerV2's scores are from (Hafner et al., 2020). SimPLe's scores are from (Kaiser et al., 2019). Go-Explore's scores are from (Ecoffet et al., 2019). Muesli's scores are from (Hessel et al., 2021). In the following, we detail the raw scores and SABER of each algorithm on 57 Atari games.

[3] 200M and 10B+ represent the training scale.

Table 17. Score table of SOTA 200M model-free algorithms on SABER(%) (GDI-I^3).

Games	RND	HWR	RAINBOW	SABER	IMPALA	SABER	LASER	SABER	GDI-I^3	SABER
Scale			200M		200M		200M		200M	
Alien	227.8	**251916**	9491.7	3.68	15962.1	6.25	976.51	14.04	43384	17.15
Amidar	5.8	**104159**	5131.2	4.92	1554.79	1.49	1829.2	1.75	1442	1.38
Assault	222.4	8647	14198.5	165.90	19148.47	200.00	21560.4	200.00	63876	200.00
Asterix	210	**1000000**	428200	42.81	300732	30.06	240090	23.99	759910	75.99
Asteroids	719	**10506650**	2712.8	0.02	108590.05	1.03	213025	2.02	751970	7.15
Atlantis	12850	**10604840**	826660	7.68	849967.5	7.90	841200	7.82	3803000	35.78
Bank Heist	14.2	**82058**	1358	1.64	1223.15	1.47	569.4	0.68	1401	1.69
Battle Zone	236	801000	62010	7.71	20885	2.58	64953.3	8.08	478830	59.77
Beam Rider	363.9	**999999**	16850.2	1.65	32463.47	3.21	90881.6	9.06	162100	16.18
Berzerk	123.7	**1057940**	2545.6	0.23	1852.7	0.16	25579.5	2.41	7607	0.71
Bowling	23.1	**300**	30	2.49	59.92	13.30	48.3	9.10	201.9	64.57
Boxing	0.1	**100**	99.6	99.60	99.96	99.96	**100**	**100.00**	**100**	**100.00**
Breakout	1.7	**864**	417.5	48.22	787.34	91.11	747.9	86.54	**864**	**100.00**
Centipede	2090.9	**1301709**	8167.3	0.47	11049.75	0.69	292792	22.37	155830	11.83
Chopper Command	811	**999999**	16654	1.59	28255	2.75	761699	76.15	**999999**	**100.00**
Crazy Climber	10780.5	219900	168788.5	75.56	136950	60.33	167820	75.10	201000	90.96
Defender	2874.5	**6010500**	55105	0.87	185203	3.03	336953	5.56	893110	14.82
Demon Attack	152.1	**1556345**	111185	7.13	132826.98	8.53	133530	8.57	675530	43.10
Double Dunk	-18.6	21	-0.3	46.21	-0.33	46.14	14	82.32	**24**	**107.58**
Enduro	0	9500	2125.9	22.38	0	0.00	0	0.00	**14330**	**150.84**
Fishing Derby	-91.7	**71**	31.3	75.60	44.85	83.93	45.2	84.14	59	95.08
Freeway	0	**38**	34	89.47	0	0.00	0	0.00	34	89.47
Frostbite	65.2	**454830**	9590.5	2.09	317.75	0.06	5083.5	1.10	10485	2.29
Gopher	257.6	355040	70354.6	19.76	66782.3	18.75	114820.7	32.29	**488830**	**137.71**
Gravitar	173	**162850**	1419.3	0.77	359.5	0.11	1106.2	0.57	5905	3.52
Hero	1027	**1000000**	55887.4	5.49	33730.55	3.27	31628.7	3.06	38330	3.73
Ice Hockey	-11.2	36	1.1	26.06	3.48	31.10	17.4	60.59	44.92	118.94
Jamesbond	29	45550	19809	43.45	601.5	1.26	37999.8	83.41	594500	200.00
Kangaroo	52	**1424600**	14637.5	1.02	1632	0.11	14308	1.00	14500	1.01
Krull	1598	104100	8741.5	6.97	8147.4	6.39	9387.5	7.60	97575	93.63
Kung Fu Master	258.5	1000000	52181	5.19	43375.5	4.31	607443	60.73	140440	14.02
Montezuma Revenge	0	**1219200**	384	0.03	0	0.00	0.3	0.00	3000	0.25
Ms Pacman	307.3	**290090**	5380.4	1.75	7342.32	2.43	6565.5	2.16	11536	3.87
Name This Game	2292.3	25220	13136	47.30	21537.2	83.94	26219.5	104.36	34434	140.19
Phoenix	761.5	**4014440**	108529	2.69	210996.45	5.24	519304	12.92	894460	22.27
Pitfall	-229.4	**114000**	**0**	**0.20**	-1.66	0.20	-0.6	0.20	**0**	0.20
Pong	-20.7	**21**	20.9	99.76	20.98	99.95	**21**	**100.00**	**21**	**100.00**
Private Eye	24.9	**101800**	4234	4.14	98.5	0.07	96.3	0.07	15100	14.81
Qbert	163.9	**2400000**	33817.5	1.40	351200.12	14.63	21449.6	0.89	27800	1.03
Riverraid	1338.5	**1000000**	22920.8	2.16	29608.05	2.83	40362.7	3.91	28075	2.68
Road Runner	11.5	**2038100**	62041	3.04	57121	2.80	45289	2.22	878600	43.11
Robotank	2.2	76	61.4	80.22	12.96	14.58	62.1	81.17	108.2	143.63
Seaquest	68.4	999999	15898.9	1.58	1753.2	0.17	2890.3	0.28	943910	94.39
Skiing	-17098	**-3272**	-12957.8	29.95	-10180.38	50.03	-29968.4	-93.09	-6774	74.67
Solaris	1236.3	**111420**	3560.3	2.11	2365	1.02	2273.5	0.94	11074	8.93
Space Invaders	148	**621535**	18789	3.00	43595.78	6.99	51037.4	8.19	140460	22.58
Star Gunner	664	77400	127029	164.67	200625	200.00	321528	418.14	465750	200.00
Surround	-10	9.6	**9.7**	**100.51**	7.56	89.59	8.4	93.88	-7.8	11.22
Tennis	-23.8	21	0	53.13	0.55	54.35	12.2	80.36	**24**	**106.70**
Time Pilot	3568	65300	12926	15.16	48481.5	72.76	105316	164.82	216770	200.00
Tutankham	11.4	**5384**	241	4.27	292.11	5.22	278.9	4.98	423.9	7.68
Up N Down	533.4	82840	125755	152.14	332546.75	200.00	345727	200.00	**986440**	**200.00**
Venture	0	**38900**	5.5	0.01	0	0.00	0	0.00	2000	5.14
Video Pinball	0	**89218328**	533936.5	0.60	572898.27	0.64	511835	0.57	925830	1.04
Wizard of Wor	563.5	**395300**	17862.5	4.38	9157.5	2.18	29059.3	7.22	64439	16.18
Yars Revenge	3092.9	**15000105**	102557	0.66	84231.14	0.54	166292.3	1.09	972000	6.46
Zaxxon	32.5	83700	22209.5	26.51	32935.5	39.33	41118	49.11	109140	130.41
MEAN SABER(%)	0.00	**100.00**		28.39		29.45		36.78		61.66
MEDIAN SABER(%)	0.00	**100.00**		4.92		4.31		8.08		35.78

Table 18. Score table of SOTA 200M model-free algorithms on SABER(%) (GDI-H^3).

Games	RND	HWR	RAINBOW	SABER	IMPALA	SABER	LASER	SABER	GDI-H^3	SABER
Scale			200M		200M		200M		200M	
Alien	227.8	**251916**	9491.7	3.68	15962.1	6.25	976.51	14.04	48735	19.27
Amidar	5.8	**104159**	5131.2	4.92	1554.79	1.49	1829.2	1.75	1065	1.02
Assault	222.4	8647	14198.5	165.90	19148.47	200.00	21560.4	200.00	**97155**	**200.00**
Asterix	210	**1000000**	428200	42.81	300732	30.06	240090	23.99	999999	100.00
Asteroids	719	**10506650**	2712.8	0.02	108590.05	1.03	213025	2.02	760005	7.23
Atlantis	12850	**10604840**	826660	7.68	849967.5	7.90	841200	7.82	3837300	36.11
Bank Heist	14.2	**82058**	1358	1.64	1223.15	1.47	569.4	0.68	1380	1.66
Battle Zone	236	801000	62010	7.71	20885	2.58	64953.3	8.08	**824360**	**102.92**
Beam Rider	363.9	**999999**	16850.2	1.65	32463.47	3.21	90881.6	9.06	422390	42.22
Berzerk	123.7	**1057940**	2545.6	0.23	1852.7	0.16	25579.5	2.41	14649	1.37
Bowling	23.1	**300**	30	2.49	59.92	13.30	48.3	9.10	205.2	65.76
Boxing	0.1	**100**	99.6	99.60	99.96	99.96	**100**	**100.00**	**100**	**100.00**
Breakout	1.7	**864**	417.5	48.22	787.34	91.11	747.9	86.54	**864**	**100.00**
Centipede	2090.9	**1301709**	8167.3	0.47	11049.75	0.69	292792	22.37	195630	14.89
Chopper Command	811	**999999**	16654	1.59	28255	2.75	761699	76.15	**999999**	**100.00**
Crazy Climber	10780.5	219900	168788.5	75.56	136950	60.33	167820	75.10	**241170**	**110.17**
Defender	2874.5	**6010500**	55105	0.87	185203	3.03	336953	5.56	970540	16.11
Demon Attack	152.1	**1556345**	111185	7.13	132826.98	8.53	133530	8.57	787985	50.63
Double Dunk	-18.6	21	-0.3	46.21	-0.33	46.14	14	82.32	**24**	**107.58**
Enduro	0	9500	2125.9	22.38	0	0.00	0	0.00	14300	150.53
Fishing Derby	-91.7	**71**	31.3	75.60	44.85	83.93	45.2	84.14	65	96.31
Freeway	0	**38**	34	89.47	0	0.00	0	0.00	34	89.47
Frostbite	65.2	**454830**	9590.5	2.09	317.75	0.06	5083.5	1.10	11330	2.48
Gopher	257.6	355040	70354.6	19.76	66782.3	18.75	114820.7	32.29	473560	133.41
Gravitar	173	**162850**	1419.3	0.77	359.5	0.11	1106.2	0.57	5915	3.53
Hero	1027	**1000000**	55887.4	5.49	33730.55	3.27	31628.7	3.06	38225	3.72
Ice Hockey	-11.2	36	1.1	26.06	3.48	31.10	17.4	60.59	**47.11**	**123.54**
Jamesbond	29	45550	19809	43.45	601.5	1.26	37999.8	83.41	**620780**	**200.00**
Kangaroo	52	**1424600**	14637.5	1.02	1632	0.11	14308	1.00	14636	1.02
Krull	1598	104100	8741.5	6.97	8147.4	6.39	9387.5	7.60	**594540**	**200.00**
Kung Fu Master	258.5	1000000	52181	5.19	43375.5	4.31	607443	60.73	**1666665**	**166.68**
Montezuma Revenge	0	**1219200**	384	0.03	0	0.00	0.3	0.00	2500	0.21
Ms Pacman	307.3	**290090**	5380.4	1.75	7342.32	2.43	6565.5	2.16	11573	3.89
Name This Game	2292.3	25220	13136	47.30	21537.2	83.94	26219.5	104.36	**36296**	**148.31**
Phoenix	761.5	**4014440**	108529	2.69	210996.45	5.24	519304	12.92	959580	23.89
Pitfall	-229.4	**114000**	**0**	**0.20**	-1.66	0.20	-0.6	0.20	-4.3	0.20
Pong	-20.7	**21**	20.9	99.76	20.98	99.95	**21**	**100.00**	**21**	**100.00**
Private Eye	24.9	**101800**	4234	4.14	98.5	0.07	96.3	0.07	15100	14.81
Qbert	163.9	**2400000**	33817.5	1.40	351200.12	14.63	21449.6	0.89	28657	1.19
Riverraid	1338.5	**1000000**	22920.8	2.16	29608.05	2.83	40362.7	3.91	28349	2.70
Road Runner	11.5	**2038100**	62041	3.04	57121	2.80	45289	2.22	999999	49.06
Robotank	2.2	76	61.4	80.22	12.96	14.58	62.1	81.17	**113.4**	**150.68**
Seaquest	68.4	999999	15898.9	1.58	1753.2	0.17	2890.3	0.28	**1000000**	**100.00**
Skiing	-17098	**-3272**	-12957.8	29.95	-10180.38	50.03	-29968.4	-93.09	-6025	86.77
Solaris	1236.3	**111420**	3560.3	2.11	2365	1.02	2273.5	0.94	9105	7.14
Space Invaders	148	**621535**	18789	3.00	43595.78	6.99	51037.4	8.19	154380	24.82
Star Gunner	664	77400	127029	164.67	200625	200.00	321528	418.14	**677590**	**200.00**
Surround	-10	9.6	**9.7**	**100.51**	7.56	89.59	8.4	93.88	2.606	64.32
Tennis	-23.8	21	0	53.13	0.55	54.35	12.2	80.36	**24**	**106.70**
Time Pilot	3568	65300	12926	15.16	48481.5	72.76	105316	164.82	**450810**	**200.00**
Tutankham	11.4	**5384**	241	4.27	292.11	5.22	278.9	4.98	418.2	7.57
Up N Down	533.4	82840	125755	152.14	332546.75	200.00	345727	200.00	966590	200.00
Venture	0	**38900**	5.5	0.01	0	0.00	0	0.00	2000	5.14
Video Pinball	0	**89218328**	533936.5	0.60	572898.27	0.64	511835	0.57	978190	1.10
Wizard of Wor	563.5	**395300**	17862.5	4.38	9157.5	2.18	29059.3	7.22	63735	16.00
Yars Revenge	3092.9	**15000105**	102557	0.66	84231.14	0.54	166292.3	1.09	968090	6.43
Zaxxon	32.5	83700	22209.5	26.51	32935.5	39.33	41118	49.11	**216020**	**200.00**
MEAN SABER(%)	0.00	**100.00**		28.39		29.45		36.78		71.26
MEDIAN SABER(%)	0.00	**100.00**		4.92		4.31		8.08		50.63

Table 19. Score table of SOTA 10B+ model-free algorithms on SABER(%).

Games	R2D2	SABER	NGU	SABER	AGENT57	SABER	GDI-I^3	SABER	GDI-H^3	SABER
Scale	10B		35B		100B		200M		200M	
Alien	109038.4	43.23	248100	98.48	**297638.17**	**118.17**	43384	17.15	48735	19.27
Amidar	27751.24	26.64	17800	17.08	**29660.08**	**28.47**	1442	1.38	1065	1.02
Assault	90526.44	200.00	34800	200.00	67212.67	200.00	63876	200.00	**97155**	**200.00**
Asterix	999080	99.91	950700	95.07	991384.42	99.14	759910	75.99	**999999**	**100.00**
Asteroids	265861.2	2.52	230500	2.19	150854.61	1.43	751970	7.15	**760005**	**7.23**
Atlantis	1576068	14.76	1653600	15.49	1528841.76	14.31	3803000	35.78	**3837300**	**36.11**
Bank Heist	**46285.6**	**56.40**	17400	21.19	23071.5	28.10	1401	1.69	1380	1.66
Battle Zone	513360	64.08	691700	86.35	**934134.88**	**116.63**	478830	59.77	824360	102.92
Beam Rider	128236.08	12.79	63600	6.33	300509.8	30.03	162100	16.18	**422390**	**42.22**
Berzerk	34134.8	3.22	36200	3.41	**61507.83**	**5.80**	7607	0.71	14649	1.37
Bowling	196.36	62.57	211.9	68.18	**251.18**	**82.37**	201.9	64.57	205.2	65.76
Boxing	99.16	99.16	99.7	99.70	**100**	**100.00**	**100**	**100.00**	**100**	**100.00**
Breakout	795.36	92.04	559.2	64.65	790.4	91.46	**864**	**100.00**	**864**	**100**
Centipede	532921.84	40.85	**577800**	44.30	412847.86	31.61	155830	11.83	195630	14.89
Chopper Command	960648	96.06	999900	99.99	999900	99.99	**999999**	**100.00**	**999999**	**100.00**
Crazy Climber	312768	144.41	313400	144.71	**565909.85**	**200.00**	201000	90.96	241170	110.17
Defender	562106	9.31	664100	11.01	677642.78	11.23	893110	14.82	**970540**	**16.11**
Demon Attack	143664.6	9.22	143500	9.21	143161.44	9.19	675530	43.10	**787985**	**50.63**
Double Dunk	23.12	105.35	-14.1	11.36	23.93	107.40	**24**	**107.58**	**24**	**107.58**
Enduro	2376.68	25.02	2000	21.05	2367.71	24.92	**14330**	**150.84**	14300	150.53
Fishing Derby	81.96	106.74	32	76.03	**86.97**	**109.82**	59	95.08	65	96.31
Freeway	**34**	**89.47**	28.5	75.00	32.59	85.76	**34**	**89.47**	**34**	**89.47**
Frostbite	11238.4	2.46	206400	45.37	**541280.88**	**119.01**	10485	2.29	11330	2.48
Gopher	122196	34.37	113400	31.89	117777.08	33.12	**488830**	**137.71**	473560	133.41
Gravitar	6750	4.04	14200	8.62	**19213.96**	**11.70**	5905	3.52	5915	3.53
Hero	37030.4	3.60	69400	6.84	**114736.26**	**11.38**	38330	3.73	38225	3.72
Ice Hockey	**71.56**	**175.34**	-4.1	15.04	63.64	158.56	44.92	118.94	**47.11**	**123.54**
Jamesbond	23266	51.05	26600	58.37	135784.96	200.00	594500	200.00	**620780**	**200.00**
Kangaroo	14112	0.99	**35100**	2.46	24034.16	1.68	14500	1.01	14636	1.02
Krull	145284.8	140.18	127400	122.73	251997.31	200.00	97575	93.63	**594540**	**200.00**
Kung Fu Master	200176	20.00	212100	21.19	206845.82	20.66	140440	14.02	**1666665**	**166.68**
Montezuma Revenge	2504	0.21	**10400**	0.85	9352.01	0.77	3000	0.25	2500	0.21
Ms Pacman	29928.2	10.22	40800	13.97	**63994.44**	**21.98**	11536	3.87	11573	3.89
Name This Game	45214.8	187.21	23900	94.24	**54386.77**	**200.00**	34434	140.19	36296	148.31
Phoenix	811621.6	20.20	959100	23.88	908264.15	22.61	894460	22.27	**959580**	**23.89**
Pitfall	**0**	**0.20**	7800	7.03	**18756.01**	**16.62**	0	0.20	-4.3	0.20
Pong	**21**	**100.00**	19.6	96.64	20.67	99.21	**21**	**100.00**	**21**	**100.00**
Private Eye	300	0.27	**100000**	98.23	79716.46	78.30	15100	14.81	15100	14.81
Qbert	161000	6.70	451900	18.82	**580328.14**	**24.18**	27800	1.03	28657	1.19
Riverraid	34076.4	3.28	36700	3.54	**63318.67**	**6.21**	28075	2.68	28349	2.70
Road Runner	498660	24.47	128600	6.31	243025.8	11.92	**878600**	**43.11**	999999	49.06
Robotank	**132.4**	**176.42**	9.1	9.35	127.32	169.54	108	143.63	113.4	150.68
Seaquest	999991.84	100.00	**1000000**	100.00	999997.63	100.00	943910	94.39	**1000000**	**100.00**
Skiing	-29970.32	-93.10	-22977.9	-42.53	**-4202.6**	**93.27**	-6774	74.67	-6025	86.77
Solaris	4198.4	2.69	4700	3.14	**44199.93**	**38.99**	11074	8.93	9105	7.14
Space Invaders	55889	8.97	43400	6.96	48680.86	7.81	140460	22.58	**154380**	**24.82**
Star Gunner	521728	200.00	414600	200.00	**839573.53**	**200.00**	465750	200.00	677590	200.00
Surround	**9.96**	**101.84**	-9.6	2.04	9.5	99.49	-7.8	11.22	2.606	64.32
Tennis	**24**	**106.70**	10.2	75.89	23.84	106.34	**24**	**106.70**	**24**	**106.70**
Time Pilot	348932	200.00	344700	200.00	405425.31	200.00	216770	200.00	**450810**	**200.00**
Tutankham	393.64	7.11	191.1	3.34	**2354.91**	**43.62**	423.9	7.68	418.2	7.57
Up N Down	542918.8	200.00	620100	200.00	623805.73	200.00	**986440**	**200.00**	966590	200.00
Venture	1992	5.12	1700	4.37	**2623.71**	**6.74**	2000	5.14	2000	5.14
Video Pinball	483569.72	0.54	965300	1.08	**992340.74**	**1.11**	925830	1.04	978190	1.10
Wizard of Wor	133264	33.62	106200	26.76	**157306.41**	**39.71**	64439	16.18	63735	16.00
Yars Revenge	918854.32	6.11	986000	6.55	**998532.37**	**6.64**	972000	6.46	968090	6.43
Zaxxon	181372	200.00	111100	132.75	**249808.9**	**200.00**	109140	130.41	216020	200.00
MEAN SABER(%)		60.43		50.47		**76.26**		61.66		71.26
MEDIAN SABER(%)		33.62		21.19		43.62		35.78		**50.63**

Table 20. Score table of SOTA model-based algorithms on SABER(%). SimPLe (Kaiser et al., 2019) and DreamerV2 (Hafner et al., 2020) haven't evaluated all 57 Atari Games in their paper. For fairness, we set the score on those games as N/A, which will not be considered when calculating the median and mean SABER.

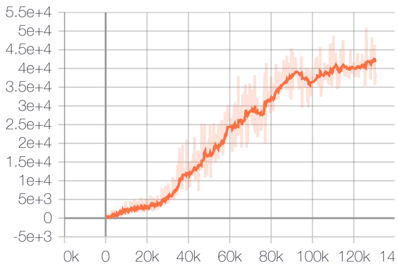
Games	MuZero	SABER	DreamerV2	SABER	SimPLe	SABER	GDI-I^3	SABER	GDI-H^3	SABER(%)
Scale	20B		200M		1M		200M		200M	
Alien	**741812.63**	**200.00**	3483	1.29	616.9	0.15	43384	17.15	48735	19.27
Amidar	**28634.39**	**27.49**	2028	1.94	74.3	0.07	1442	1.38	1065	1.02
Assault	**143972.03**	**200.00**	7679	88.51	527.2	3.62	63876	200.00	97155	200.00
Asterix	998425	99.84	25669	2.55	1128.3	0.09	759910	75.99	**999999**	**100.00**
Asteroids	678558.64	6.45	3064	0.02	793.6	0.00	751970	7.15	**760005**	**7.23**
Atlantis	1674767.2	15.69	989207	9.22	20992.5	0.08	3803000	35.78	**3837300**	**36.11**
Bank Heist	1278.98	1.54	1043	1.25	34.2	0.02	**1401**	**1.69**	1380	1.66
Battle Zone	**848623**	**105.95**	31225	3.87	4031.2	0.47	478830	59.77	824360	102.92
Beam Rider	**454993.53**	**45.48**	12413	1.21	621.6	0.03	162100	16.18	422390	42.22
Berzerk	**85932.6**	**8.11**	751	0.06	N/A	N/A	7607	0.71	14649	1.37
Bowling	**260.13**	**85.60**	48	8.99	30	2.49	202	64.57	205.2	65.76
Boxing	**100**	**100.00**	87	86.99	7.8	7.71	**100**	**100.00**	**100**	**100.00**
Breakout	**864**	**100.00**	350	40.39	16.4	1.70	**864**	**100.00**	**864**	100.00
Centipede	**1159049.27**	**89.02**	6601	0.35	N/A	N/A	155830	11.83	195630	14.89
Chopper Command	991039.7	99.10	2833	0.20	979.4	0.02	**999999**	**100.00**	**999999**	**100.00**
Crazy Climber	**458315.4**	**200.00**	141424	62.47	62583.6	24.77	201000	90.96	241170	110.17
Defender	839642.95	13.93	N/A	N/A	N/A	N/A	893110	14.82	**970540**	**16.11**
Demon Attack	143964.26	9.24	2775	0.17	208.1	0.00	675530	43.40	**787985**	**50.63**
Double Dunk	23.94	107.42	22	102.53	N/A	N/A	**24**	**107.58**	**24**	**107.58**
Enduro	2382.44	25.08	2112	22.23	N/A	N/A	**14330**	**150.84**	14300	150.53
Fishing Derby	**91.16**	**112.39**	93.24	200.00	-90.7	0.61	59	92.89	65	96.31
Freeway	33.03	86.92	**34**	**89.47**	16.7	43.95	**34**	**89.47**	**34**	**89.47**
Frostbite	**631378.53**	**138.82**	15622	3.42	236.9	0.04	10485	2.29	11330	2.48
Gopher	130345.58	36.67	53853	15.11	596.8	0.10	**488830**	**137.71**	473560	133.41
Gravitar	**6682.7**	**4.00**	3554	2.08	173.4	0.00	5905	3.52	5915	3.53
Hero	**49244.11**	**4.83**	30287	2.93	2656.6	0.16	38330	3.73	38225	3.72
Ice Hockey	**67.04**	**165.76**	29	85.17	-11.6	-0.85	44.92	118.94	47.11	123.54
Jamesbond	41063.25	90.14	9269	20.30	100.5	0.16	594500	200.00	**620780**	**200.00**
Kangaroo	**16763.6**	**1.17**	11819	0.83	51.2	0.00	14500	1.01	14636	1.02
Krull	269358.27	200.00	9687	7.89	2204.8	0.59	97575	93.63	**594540**	**200.00**
Kung Fu Master	**204824**	**20.46**	66410	6.62	14862.5	1.46	140440	14.02	1666665	166.68
Montezuma Revenge	0	0.00	1932	0.16	N/A	N/A	**3000**	**0.25**	2500	0.21
Ms Pacman	**243401.1**	**83.89**	5651	1.84	1480	0.40	11536	3.87	11573	3.89
Name This Game	**157177.85**	**200.00**	14472	53.12	2420.7	0.56	34434	140.19	36296	148.31
Phoenix	**955137.84**	**23.78**	13342	0.31	N/A	N/A	894460	22.27	959580	23.89
Pitfall	**0**	**0.20**	-1	0.20	N/A	N/A	**0**	**0.20**	-4.3	0.20
Pong	**21**	**100.00**	19	95.20	12.8	80.34	**21**	**100.00**	**21**	**100.00**
Private Eye	**15299.98**	**15.01**	158	0.13	35	0.01	15100	14.81	15100	14.81
Qbert	**72276**	**3.00**	162023	6.74	1288.8	0.05	27800	1.15	28657	1.19
Riverraid	**323417.18**	**32.25**	16249	1.49	1957.8	0.06	28075	2.68	28349	2.70
Road Runner	613411.8	30.10	88772	4.36	5640.6	0.28	878600	43.11	**999999**	**49.06**
Robotank	**131.13**	**174.70**	65	85.09	N/A	N/A	108	143.63	113.4	150.68
Seaquest	999976.52	100.00	45898	4.58	683.3	0.06	943910	94.39	**1000000**	**100.00**
Skiing	**-29968.36**	**-93.09**	-8187	64.45	N/A	N/A	-6774	74.67	-6025	86.77
Solaris	56.62	-1.07	883	-0.32	N/A	N/A	**11074**	**8.93**	9105	7.14
Space Invaders	74335.3	11.94	2611	0.40	N/A	N/A	140460	22.58	**154380**	**24.82**
Star Gunner	549271.7	200.00	29219	37.21	N/A	N/A	465750	200.00	**677590**	**200.00**
Surround	**9.99**	**101.99**	N/A	N/A	N/A	N/A	-8	11.22	2.606	64.32
Tennis	0	53.13	23	104.46	N/A	N/A	**24**	**106.70**	**24**	**106.70**
Time Pilot	**476763.9**	**200.00**	32404	46.71	N/A	N/A	216770	200.00	450810	200.00
Tutankham	**491.48**	**8.94**	238	4.22	N/A	N/A	424	7.68	418.2	7.57
Up N Down	715545.61	200.00	648363	200.00	3350.3	3.42	**986440**	**200.00**	966590	200.00
Venture	0.4	0.00	0	0.00	N/A	N/A	**2000**	**5.23**	2000	5.14
Video Pinball	**981791.88**	**1.10**	22218	0.02	N/A	N/A	925830	1.04	978190	1.10
Wizard of Wor	**197126**	**49.80**	14531	3.54	N/A	N/A	64439	16.14	63735	16.00
Yars Revenge	553311.46	3.67	20089	0.11	5664.3	0.02	**972000**	**6.46**	968090	6.43
Zaxxon	**725853.9**	**200.00**	18295	21.83	N/A	N/A	109140	130.41	216020	200.00
MEAN SABER(%)		**71.94**		27.73		4.80		61.66		71.26
MEDIAN SABER(%)		49.80		4.29		0.13		35.78		**50.63**

Table 21. Score table of other SOTA algorithms on SABER(%). Go-Explore (Ecoffet et al., 2019) and Muesli (Hessel et al., 2021).

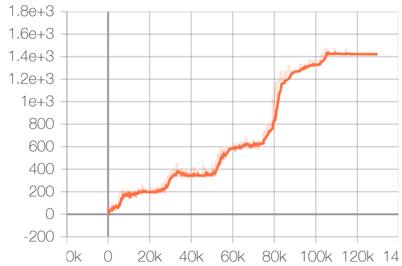
Games	Muesli	SABER	Go-Explore	SABER	GDI-I^3	SABER	GDI-H^3	SABER
Scale	200M		10B		200M		200M	
Alien	139409	55.30	**959312**	**200.00**	43384	17.15	48735	19.27
Amidar	**21653**	**20.78**	19083	18.32	1442	1.38	1065	1.02
Assault	36963	200.00	30773	200.00	63876	200.00	**97155**	**200.00**
Asterix	316210	31.61	999500	99.95	759910	75.99	**999999**	**100.00**
Asteroids	484609	4.61	112952	1.07	751970	7.15	**760005**	**7.23**
Atlantis	1363427	12.75	286460	2.58	3803000	35.78	**3837300**	**36.11**
Bank Heist	1213	1.46	**3668**	**4.45**	1401	1.69	1380	1.66
Battle Zone	414107	51.68	**998800**	**124.70**	478830	59.77	824360	102.92
Beam Rider	288870	28.86	371723	37.15	162100	16.18	**422390**	**42.22**
Berzerk	44478	4.19	**131417**	**12.41**	7607	0.71	14649	1.37
Bowling	191	60.64	**247**	**80.86**	202	64.57	205.2	65.76
Boxing	99	99.00	91	90.99	**100**	**100.00**	**100**	**100.00**
Breakout	791	91.53	774	89.56	**864**	**100.00**	**864**	**100.00**
Centipede	**869751**	**66.76**	613815	47.07	155830	11.83	195630	14.89
Chopper Command	101289	10.06	996220	99.62	**999999**	**100.00**	**999999**	**100.00**
Crazy Climber	175322	78.68	235600	107.51	201000	90.96	**241170**	**110.17**
Defender	629482	10.43	N/A	N/A	893110	14.82	**970540**	**16.11**
Demon Attack	129544	8.31	239895	15.41	675530	43.40	**787985**	**50.63**
Double Dunk	-3	39.39	**24**	**107.58**	**24**	**107.58**	**24**	**107.58**
Enduro	2362	24.86	1031	10.85	**14330**	**150.84**	14300	150.53
Fishing Derby	51	87.71	**67**	**97.54**	59	92.89	65	96.31
Freeway	33	86.84	**34**	**89.47**	**34**	**89.47**	**34**	**89.47**
Frostbite	301694	66.33	**999990**	**200.00**	10485	2.29	11330	2.48
Gopher	104441	29.37	134244	37.77	**488830**	**137.71**	473560	133.41
Gravitar	11660	7.06	**13385**	**8.12**	5905	3.52	5915	3.53
Hero	37161	3.62	37783	3.68	38330	3.73	**38225**	**3.72**
Ice Hockey	25	76.69	33	93.64	44.92	118.94	**47.11**	**123.54**
Jamesbond	19319	42.38	200810	200.00	594500	200.00	**620780**	**200.00**
Kangaroo	14096	0.99	**24300**	**1.70**	14500	1.01	14636	1.02
Krull	34221	31.83	63149	60.05	97575	93.63	**594540**	**200.00**
Kung Fu Master	134689	13.45	24320	2.41	140440	14.02	**1666665**	**166.68**
Montezuma Revenge	2359	0.19	**24758**	**2.03**	3000	0.25	2500	0.21
Ms Pacman	65278	22.42	**456123**	**157.30**	11536	3.87	11573	3.89
Name This Game	105043	200.00	**212824**	**200.00**	34434	140.19	36296	148.31
Phoenix	805305	20.05	19200	0.46	894460	22.27	**959580**	**23.89**
Pitfall	0	0.20	**7875**	**7.09**	0	0.2	-4.3	0.20
Pong	20	97.60	**21**	**100.00**	**21**	**100**	**21**	**100.00**
Private Eye	10323	10.12	**69976**	**68.73**	15100	14.81	15100	14.81
Qbert	157353	6.55	**999975**	**41.66**	27800	1.15	28657	1.19
Riverraid	**47323**	**4.60**	35588	3.43	28075	2.68	28349	2.70
Road Runner	327025	16.05	999900	49.06	878600	43.11	**999999**	**49.06**
Robotank	59	76.96	**143**	**190.79**	108	143.63	113.4	150.68
Seaquest	815970	81.60	539456	53.94	943910	94.39	**1000000**	**100.00**
Skiing	-18407	-9.47	**-4185**	**93.40**	-6774	74.67	-6025	86.77
Solaris	3031	1.63	**20306**	**17.31**	11074	8.93	9105	7.14
Space Invaders	59602	9.57	93147	14.97	140460	22.58	**154380**	**24.82**
Star Gunner	214383	200.00	609580	200.00	465750	200.00	**677590**	**200.00**
Surround	**9**	**96.94**	N/A	N/A	-8	11.22	2.606	64.32
Tennis	12	79.91	**24**	**106.7**	**24**	**106.70**	**24**	**106.70**
Time Pilot	359105	200.00	183620	200.00	216770	200.00	**450810**	**200.00**
Tutankham	252	4.48	**528**	**9.62**	424	7.68	418.2	7.57
Up N Down	649190	200.00	553718	200.00	**986440**	**11.9785**	966590	200.00
Venture	2104	5.41	**3074**	**7.90**	2035	5.23	2000	5.14
Video Pinball	685436	0.77	**999999**	**1.12**	925830	1.04	978190	1.10
Wizard of Wor	93291	23.49	**199900**	**50.50**	64293	16.14	63735	16.00
Yars Revenge	557818	3.70	**999998**	**6.65**	972000	6.46	968090	6.43
Zaxxon	65325	78.04	18340	21.88	109140	130.41	**216020**	**200.00**
MEAN SABER(%)		48.74		**71.80**		61.66		71.26
MEDIAN SABER(%)		24.86		50.50		35.78		**50.63**

K.8. Atari Games Learning Curves

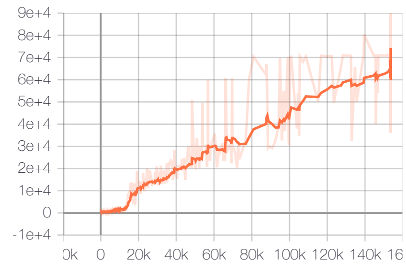
K.8.1. Atari Games Learning Curves of GDI-I[3]



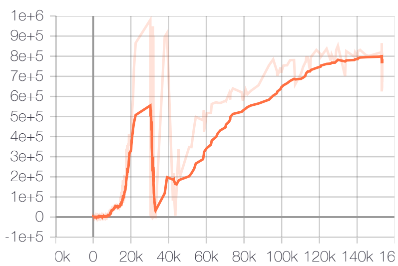
1. Alien



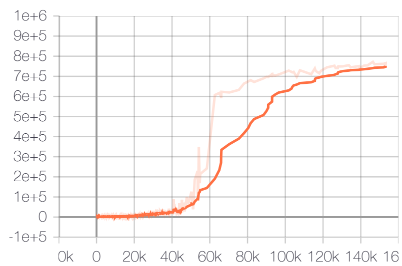
2. Amidar



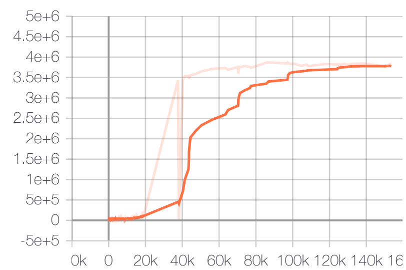
3. Assault



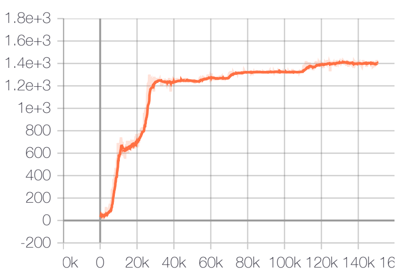
4. Asterix



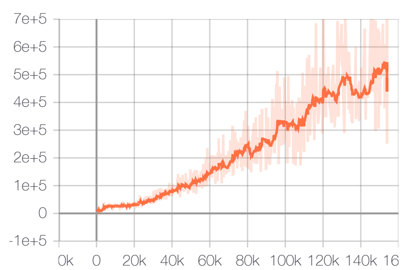
5. Asteroids



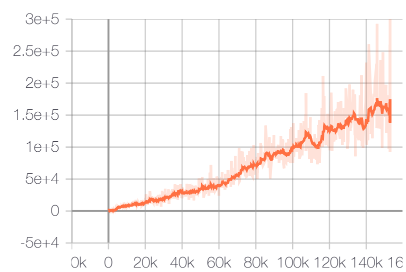
6. Atlantis



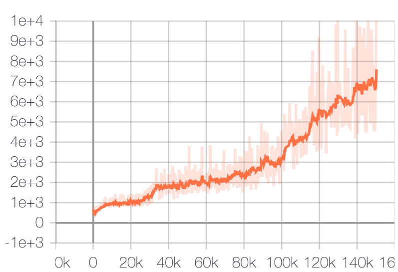
7. Bank_Heist



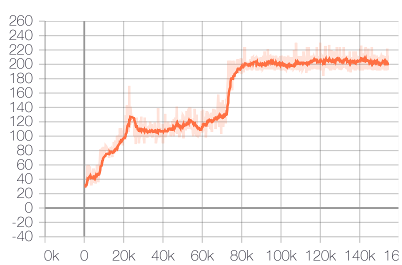
8. Battle_Zone



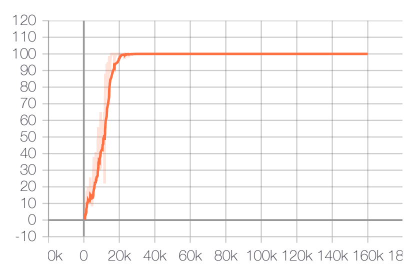
9. Beam_Rider



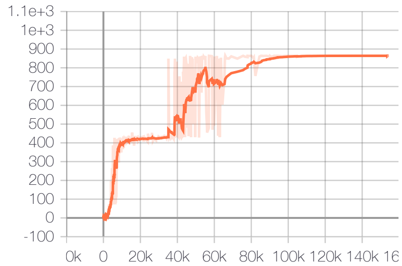
10. Berzerk



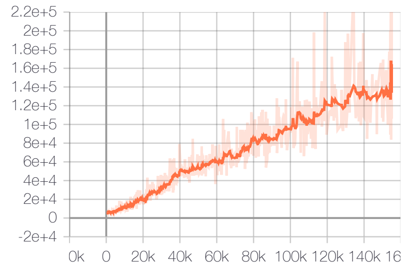
11. Bowling



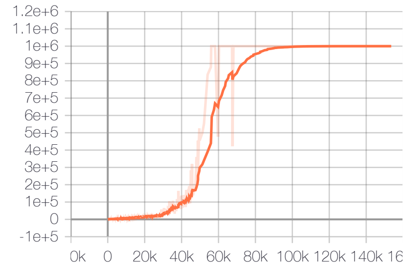
12. Boxing



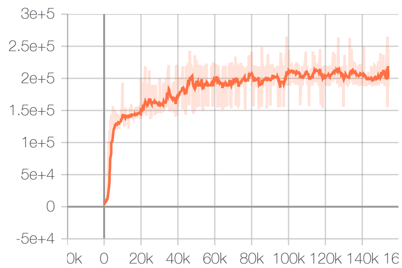
13. Breakout



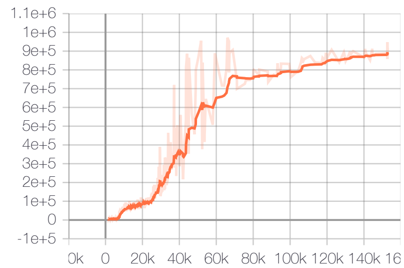
14. Centipede



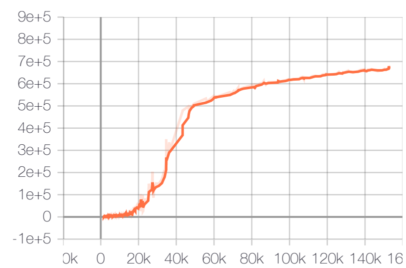
15. Chopper_Command



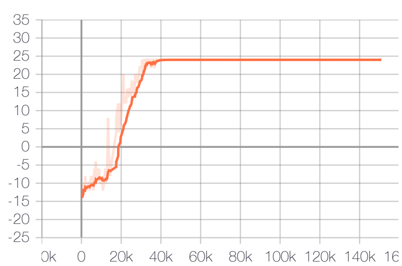
16. Crazy_Climber



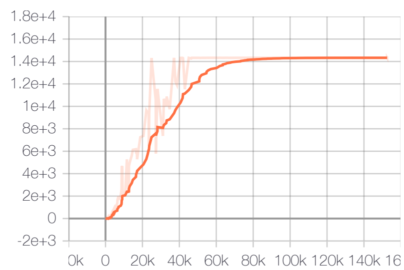
17. Defender



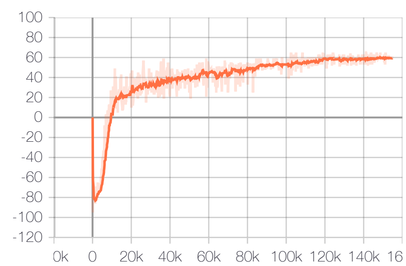
18. Demon_Attack



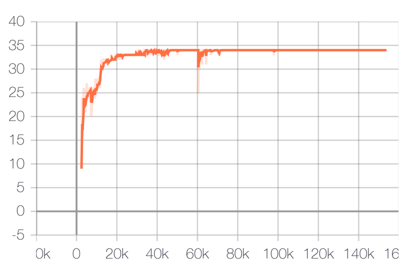
19. Double_Dunk



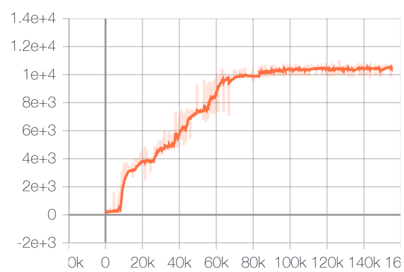
20. Enduro



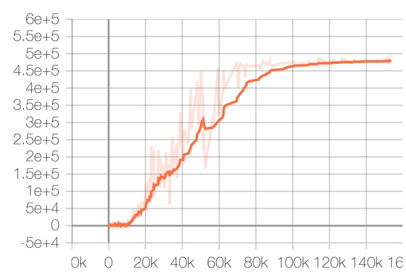
21. Fishing_Derby



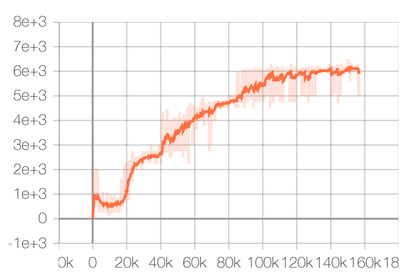
22. Freeway



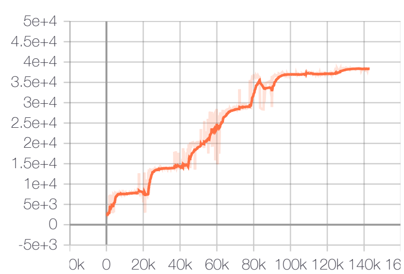
23. Frostbite



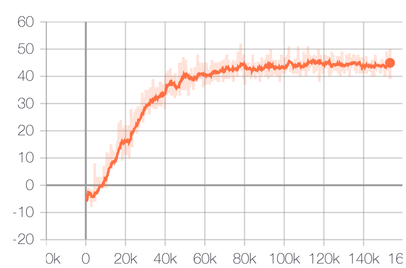
24. Gopher



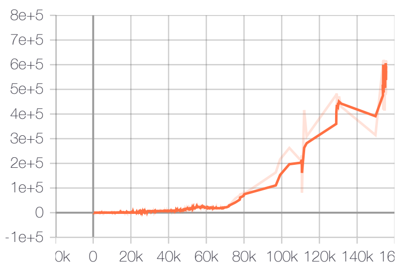
25. Gravitar



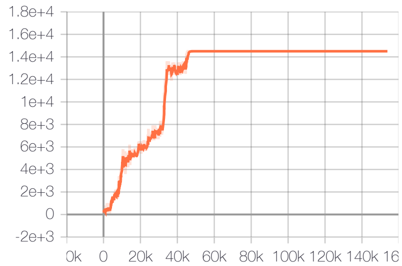
26. Hero



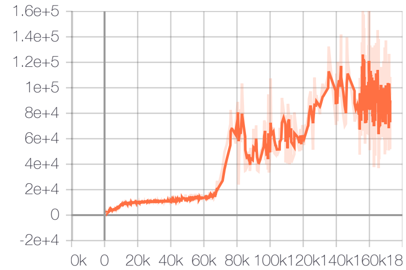
27. Ice_Hockey



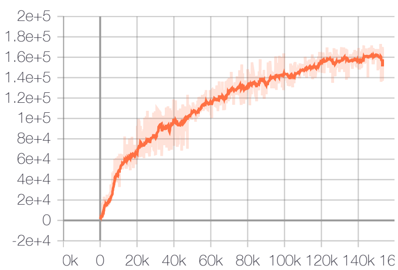
28. Jamesbond



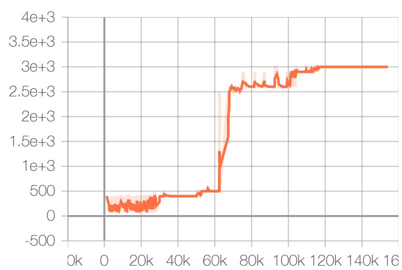
29. Kangaroo



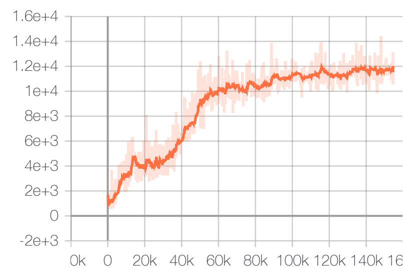
30. Krull



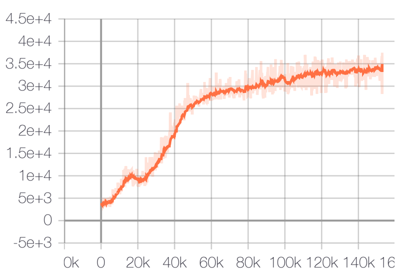
31. Kung_Fu_Master



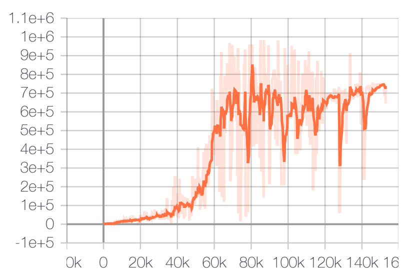
32. Montezuma_Revenge



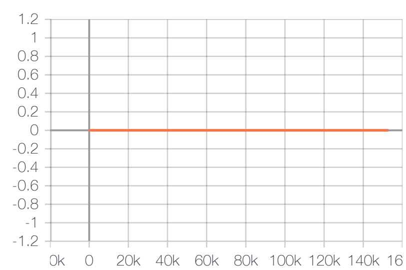
33. Ms_Pacman



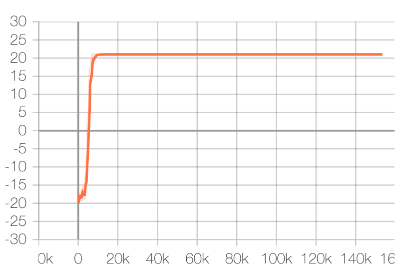
34. Name_This_Game



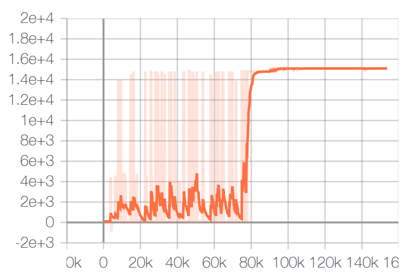
35. Phoenix



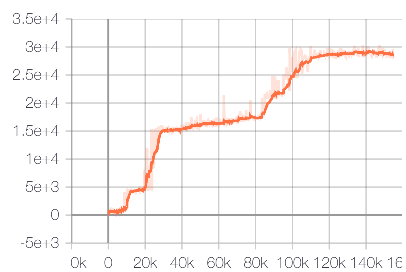
36. Pitfall



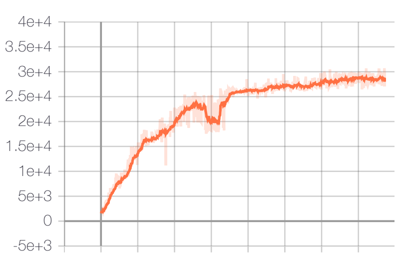
37. Pong



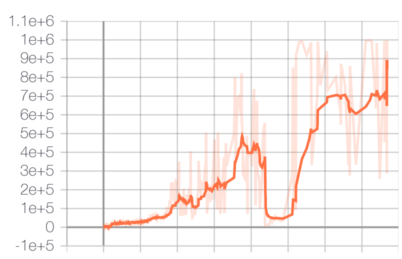
38. Private_Eye



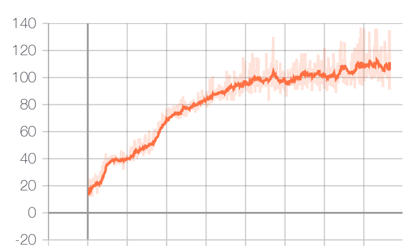
39. Qbert



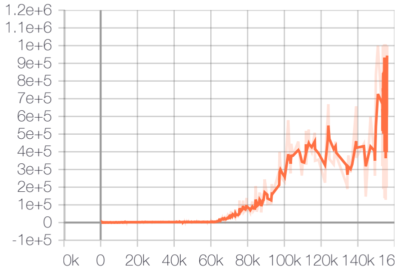
40. Riverraid



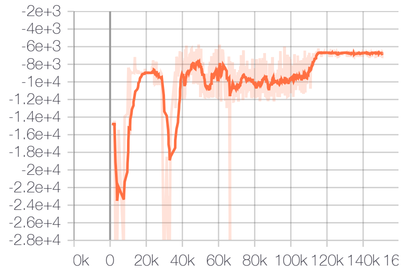
41. Road_Runner



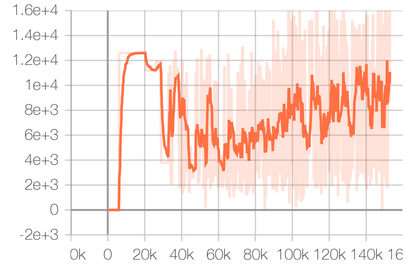
42. Robotank



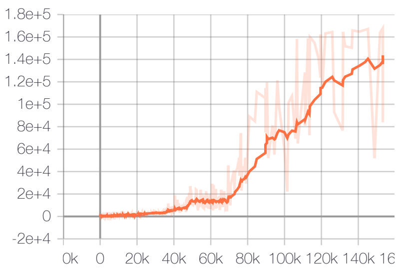
43. Seaquest



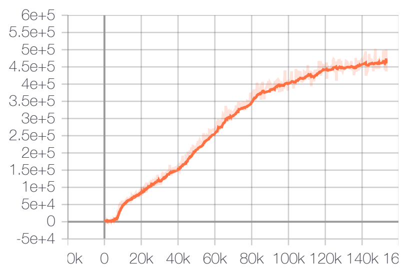
44. Skiing



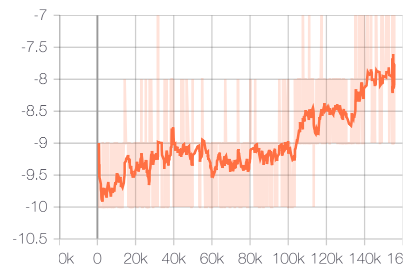
45. Solaris



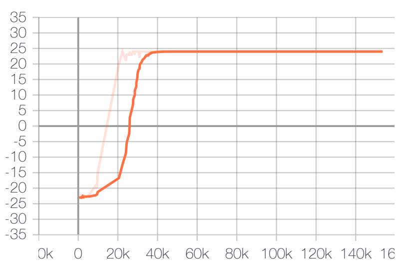
46. Space_Invaders



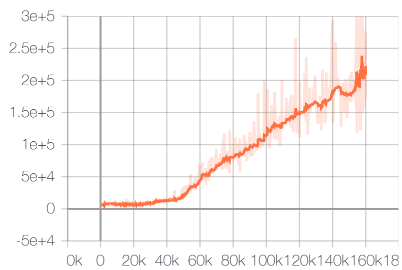
47. Star_Gunner



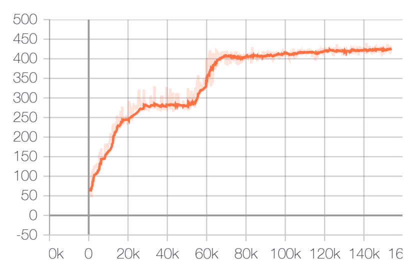
48. Surround



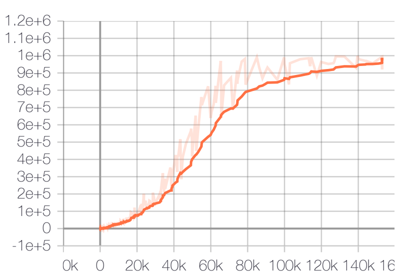
49. Tennis



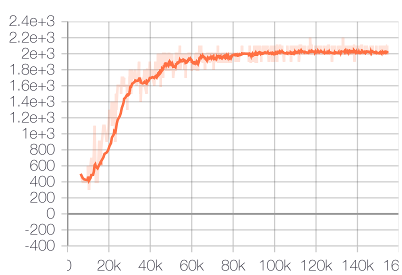
50. Time_Pilot



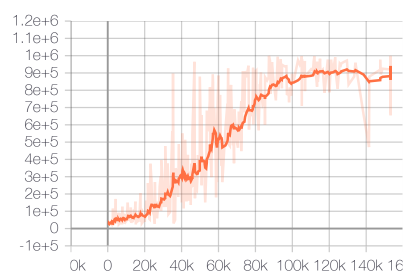
51. Tutankham



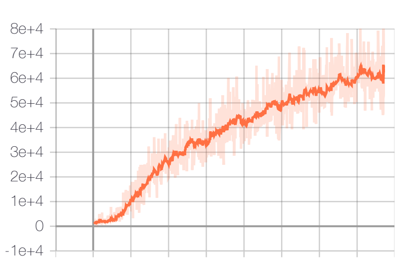
52. Up_N_Down



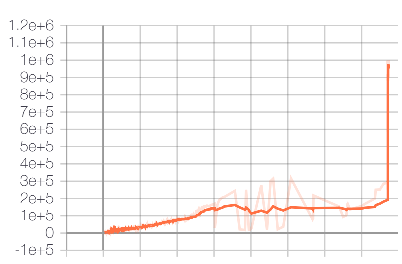
53. Venture



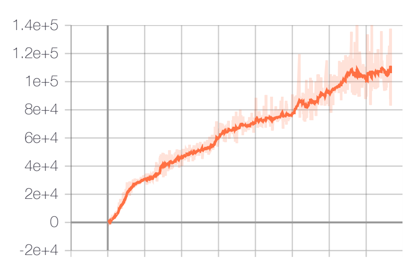
54. Video_Pinball



55. Wizard_of_Wor

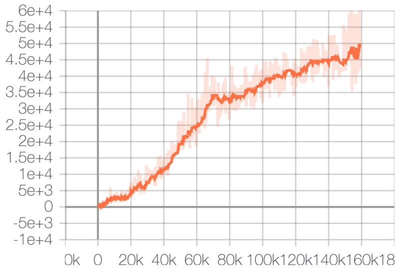


56. Yars_Revenge

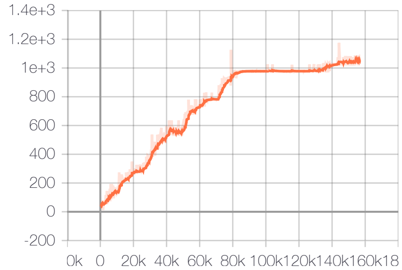


57. Zaxxon

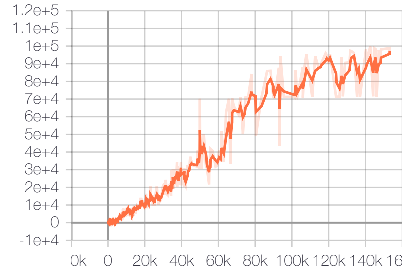
K.8.2. Atari Games Learning Curves of GDI-H^3



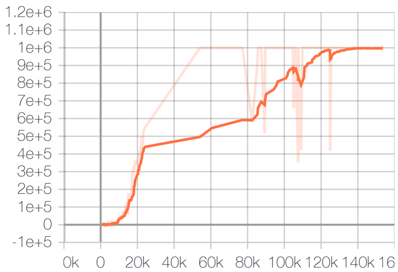
1. alien



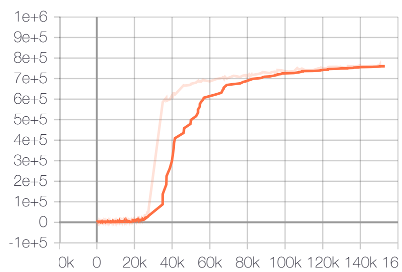
2. amidar



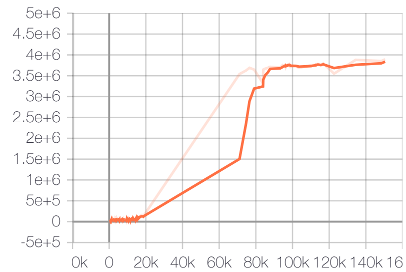
3. assault



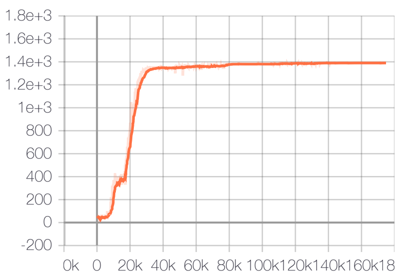
4. asterix



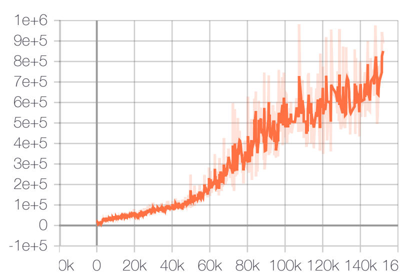
5. asteroids



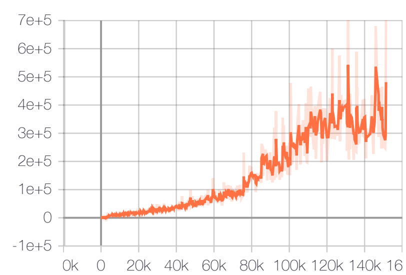
6. atlantis



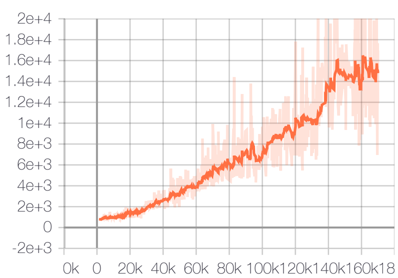
7. bank_heist



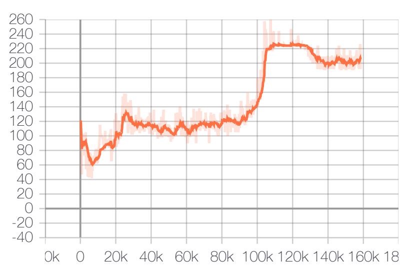
8. battle_zone



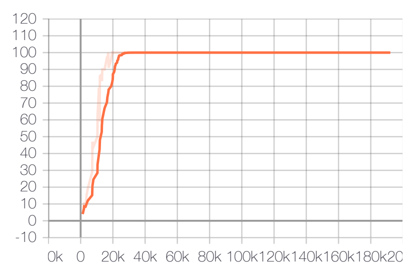
9. beam_rider



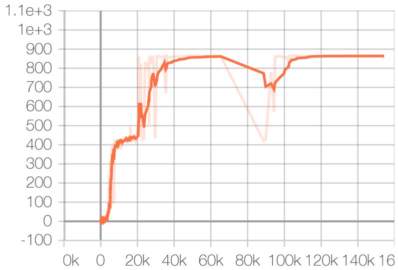
10. berzerk



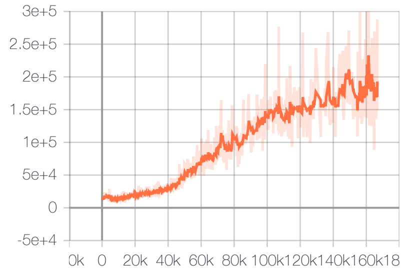
11. bowling



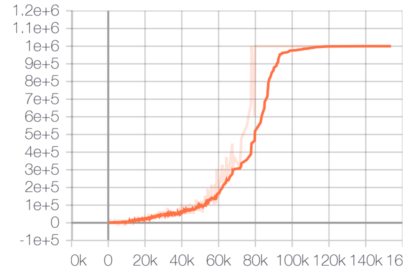
12. boxing



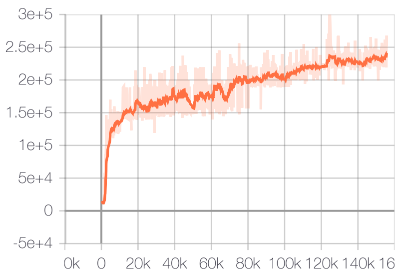
13. breakout



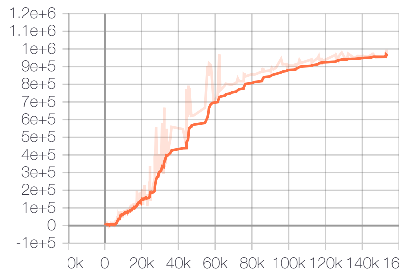
14. centipede



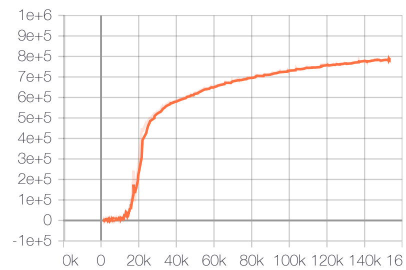
15. chopper_command



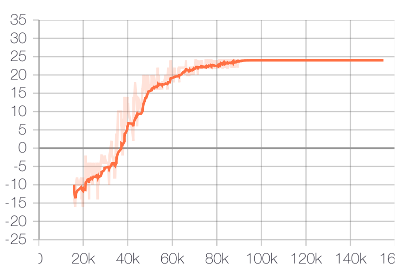
16. crazy_climber



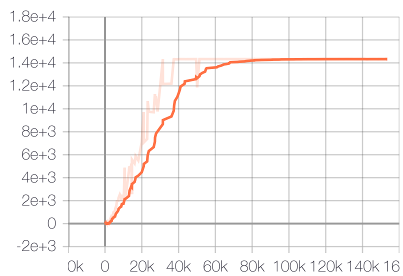
17. defender



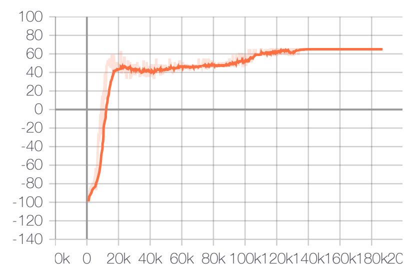
18. demon_attack



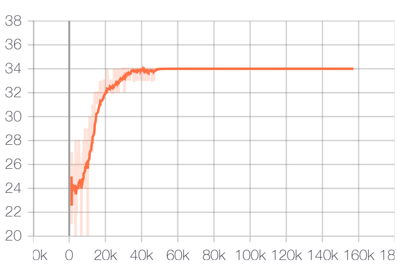
19. double_dunk



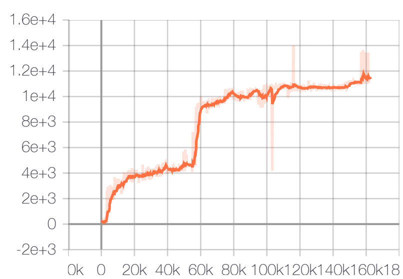
20. enduro



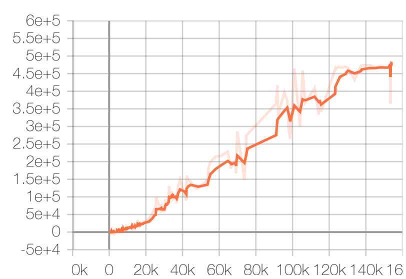
21. fishing_derby



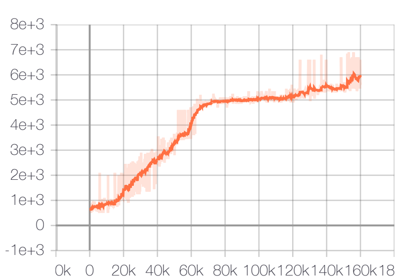
22. freeway



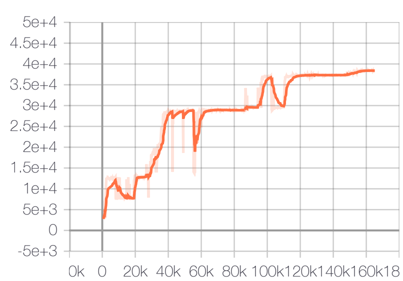
23. frostbite



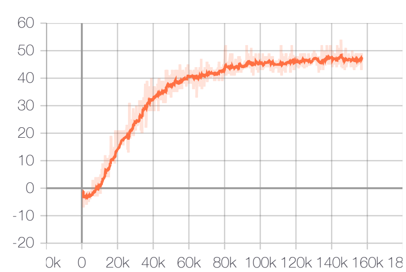
24. gopher



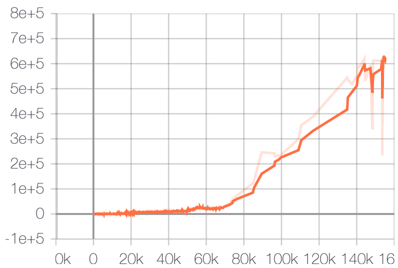
25. gravitar



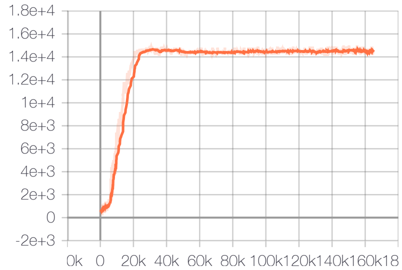
26. hero



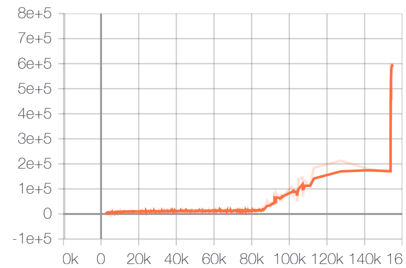
27. ice_hockey



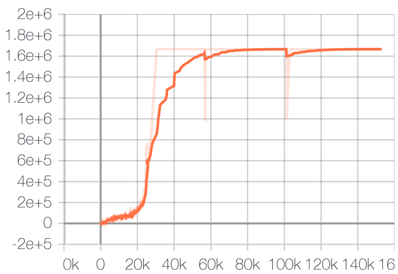
28. jamesbond



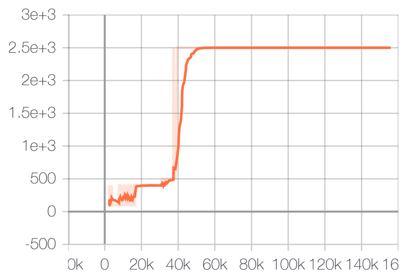
29. kangaroo



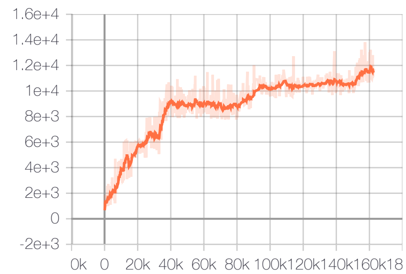
30. krull



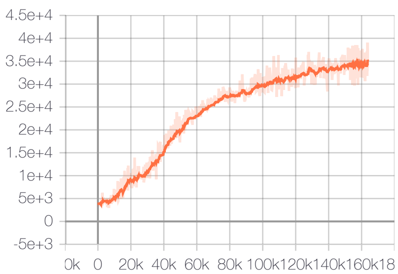
31. kung_fu_master



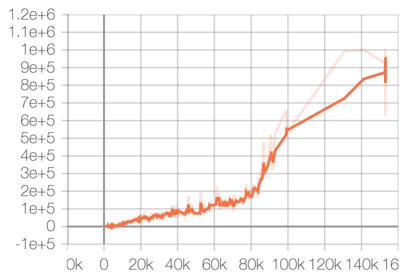
32. montezuma_revenge



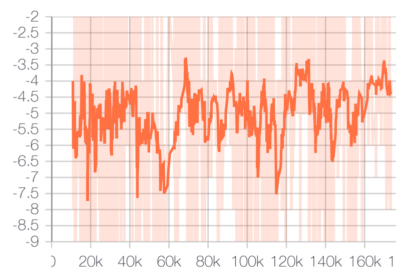
33. ms_pacman



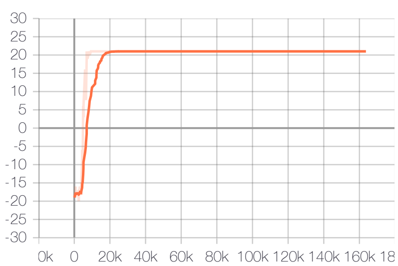
34. name_this_game



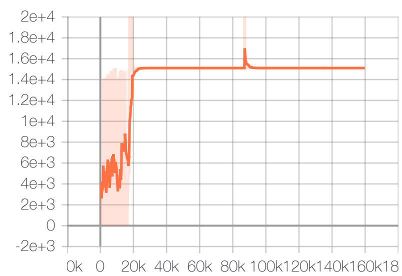
35. phoenix



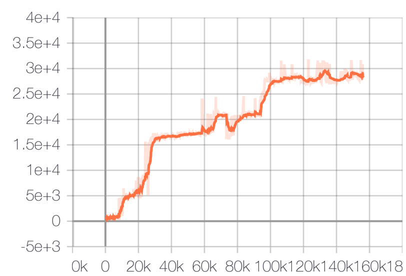
36. pitfall



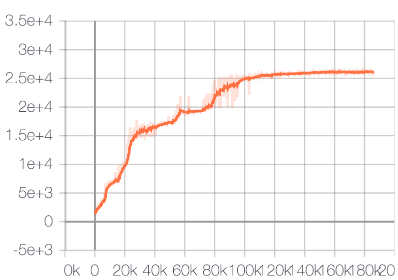
37. pong



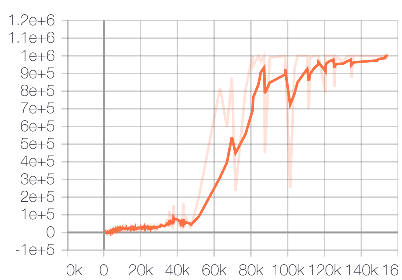
38. private_eye



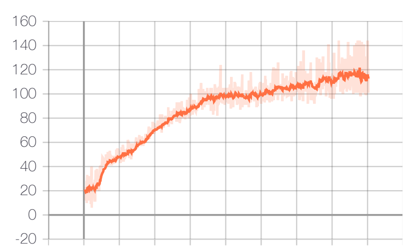
39. qbert



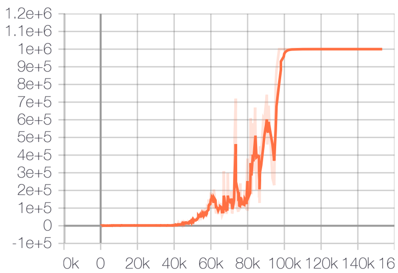
40. riverraid



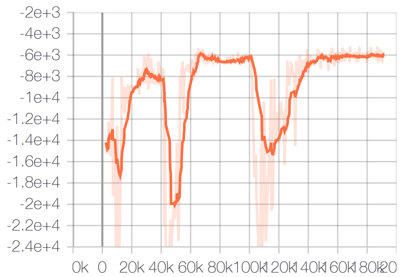
41. road_runner



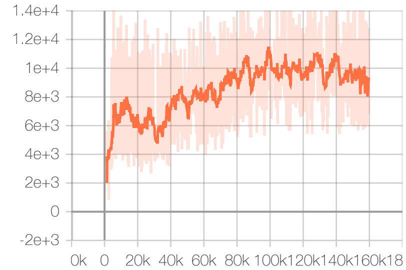
42. robotank



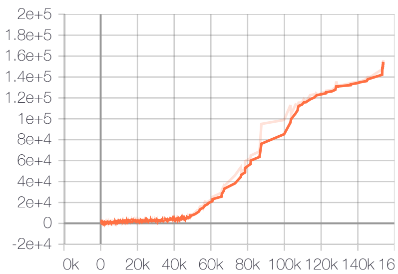
43. seaquest



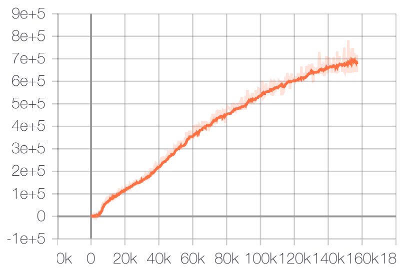
44. skiing



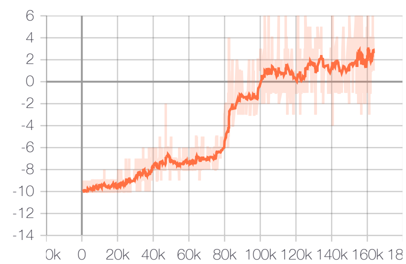
45. solaris



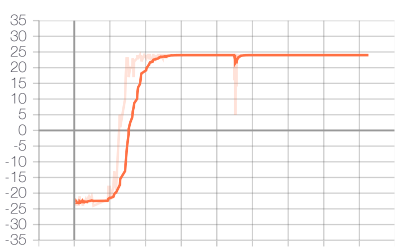
46. space_invaders



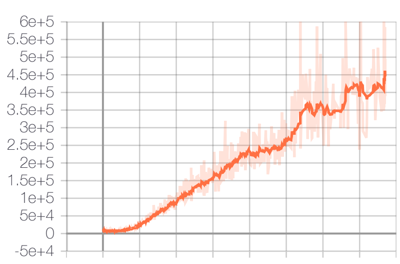
47. star_gunner



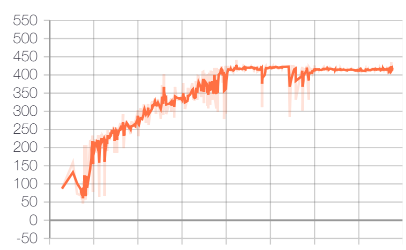
48. surround



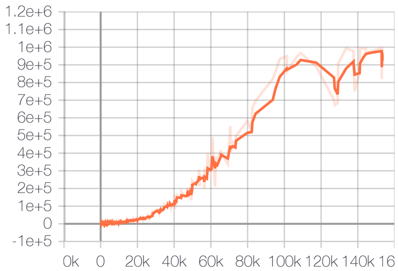
49. tennis



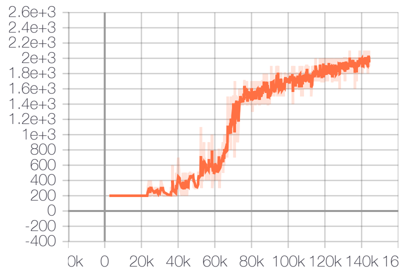
50. time_pilot



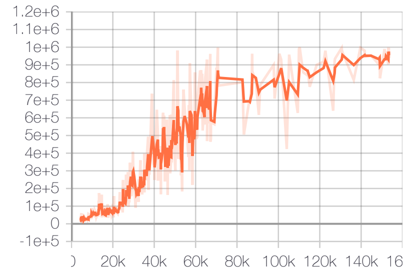
51. tutankham



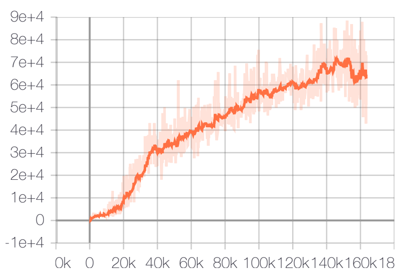
52. up_n_down



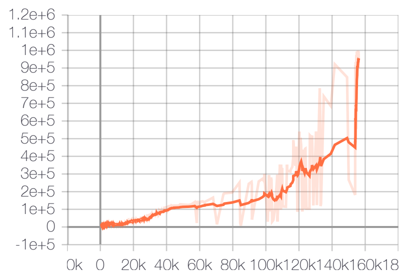
53. venture



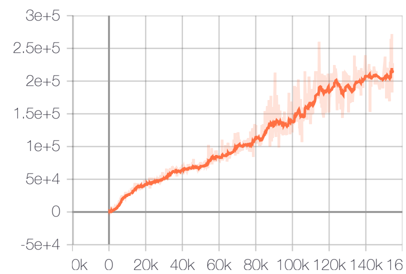
54. video_pinball



55. wizard_of_wor



56. yars_revenge



57. zaxxon

L. Ablation Study

In this section, we firstly demonstrate the settings of our ablation studies. Then, we offer the t-SNE of three Atari games as a study case to further show the data richness among different capacities of the policy space via t-SNE.

Table 22. Summary of Algorithms of Ablation Study. The behavior policies are sampled from the policy space $\{\pi_{\theta_\lambda}|\lambda \in \Lambda\}$ which is parameterized by the policy network and indexed by the index set Λ via a sampling distribution P_Λ. Wherein the sampling distribution is iteratively optimized via a data distribution optimization $\mathcal{E}$.

Name	Category	π_{θ_λ}	Λ	$P_\Lambda^{(0)}$	$\mathcal{E}$
GDI-I^3	GDI-I^3	$\epsilon \cdot \text{Softmax}\left(\frac{A_\theta}{\tau_1}\right) + (1-\epsilon) \cdot \text{Softmax}\left(\frac{A_\theta}{\tau_2}\right)$	$\{\lambda\|\lambda = (\epsilon, \tau_1, \tau_2)\}$	Uniform	MAB
GDI-H^3	GDI-H^3	$\epsilon \cdot \text{Softmax}\left(\frac{A_{\theta_1}}{\tau_1}\right) + (1-\epsilon) \cdot \text{Softmax}\left(\frac{A_{\theta_2}}{\tau_2}\right)$	$\{\lambda\|\lambda = (\epsilon, \tau_1, \tau_2)\}$	Uniform	MAB
GDI-I^0 w/o $\mathcal{E}$	GDI-I^0	$\epsilon \cdot \text{Softmax}\left(\frac{A_\theta}{\tau_1}\right) + (1-\epsilon) \cdot \text{Softmax}\left(\frac{A_\theta}{\tau_2}\right)$	$\{\lambda\|\lambda = (\epsilon, \tau_1, \tau_2)\}$	One Point	Identical Mapping
GDI-I^1	GDI-I^1	$\text{Softmax}\left(\frac{A_\theta}{\tau}\right)$	$\{\lambda\|\lambda = (\tau)\}$	Uniform	MAB

Table 23. Summary of the ablation groups in the Ablation Study. The corresponding algorithms can see Tab. L. We investigate the effects of several properties of GDI via ablating the Ablation Variables (e.g., removing the meta-controller from GDI-I^3, and explore the impact of the meta-controller) and keeping the Control Variables (e.g., Hyperparameters) remains the same.

Group Name	Ablation Variable	Control Variable	Corresponding Algorithm	Corresponding Problem	Corresponding Results
GDI-I^3	N/A	π_θ, Λ , $\mathcal{E}$	GDI-I^3	Baseline Group	N/A
w/o $\mathcal{E}$	$\mathcal{E}$	Λ, π_θ	GDI-I^0 w/o $\mathcal{E}$	Exploration-Exploitation Trade-off	Fig. 3
GDI-I^1	Λ	π_θ, $\mathcal{E}$	GDI-I^1	Data Richness	Fig. 3 Fig. 20 Fig. 21 Fig. 22 Fig. 23
GDI-H^3	π_θ	Λ , $\mathcal{E}$	GDI-H^3	Data Richness	Fig. 3 Fig. 22 Fig. 23

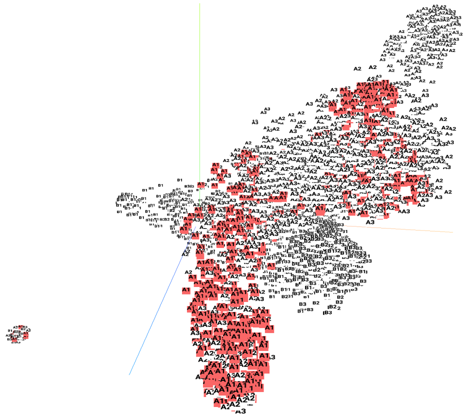
L.1. Ablation Study Design

To prove the effectiveness of capacity and diversity control and the data distribution optimization operator $\mathcal{E}$. All the implemented algorithms in the ablation study have been summarized in Tab. L.

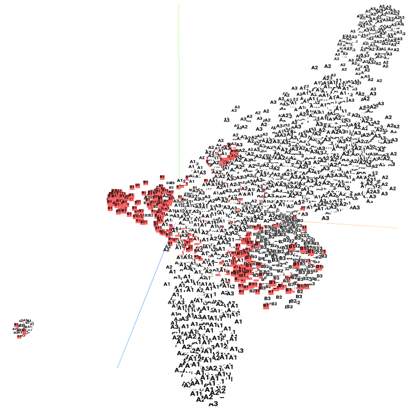
We have summarized all the ablation experimental groups of the ablation study in Tab. L. The operator $\mathcal{E}$ is achieved with MAB (see App. E).The operator $\mathcal{T}$ is achieved with Vtrace, Retrace and policy gradient. Except for the Control Variable listed in the Tab. L, other settings and the shared hyperparameters remain the same in all ablation groups. The hyperparameters can see App. G.

L.2. t-SNE

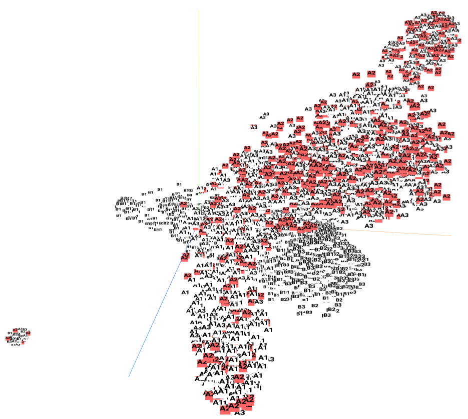
In all the t-SNE, we mark the state generated by GDI-I^3 as A$_i$ and mark the state generated by GDI-I^1 as B$_i$, where i = 1, 2, 3 represents three stages of the training process.



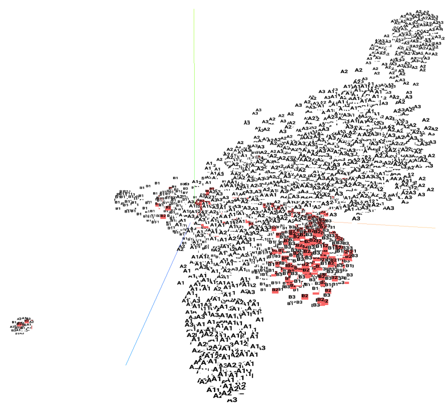
1. Early stage of GDI-I^3



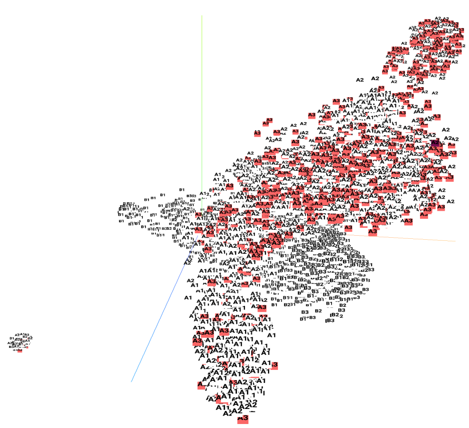
2. Early stage of GDI-I^1



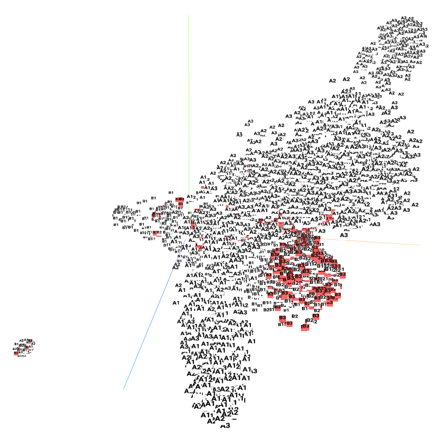
3. Middle stage of GDI-I^3



4. Middle stage of GDI-I^1

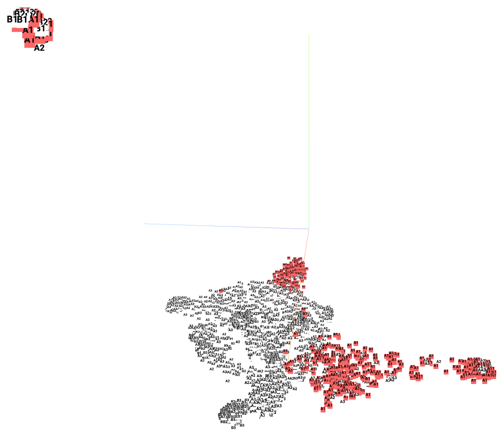


5. Later stage of GDI-I^3

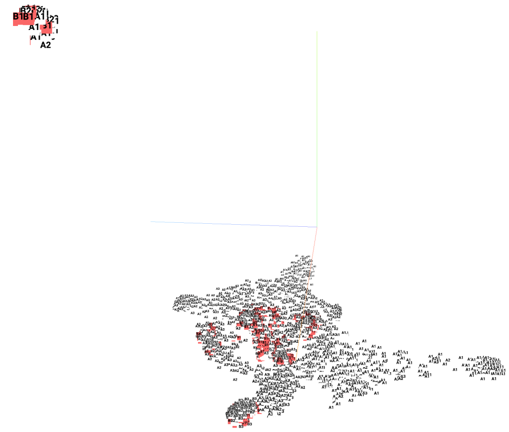


6. Later stage of GDI-I^1

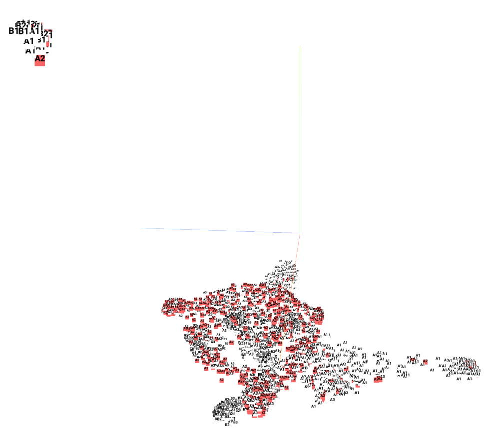
Figure 20. t-SNE of Seaquest. t-SNE is drawn from 6k states. We sample 1k states from each stage of GDI-I^3 and GDI-I^1. We highlight 1k states of each stage of GDI-I^3 and GDI-I^1.



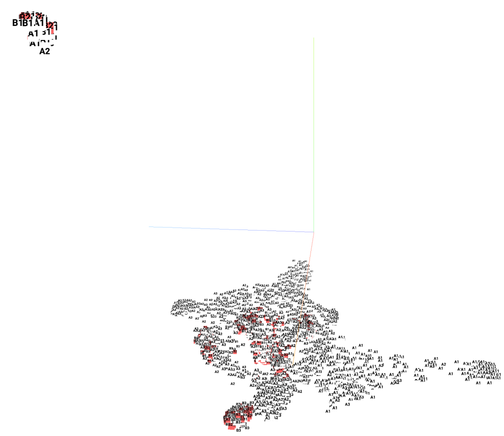
1. Early stage of GDI-I^3



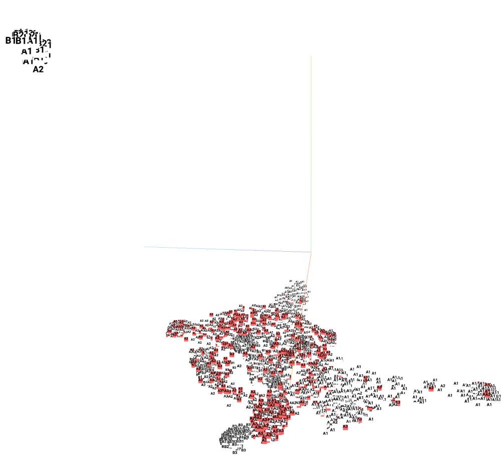
2. Early stage of GDI-I^1



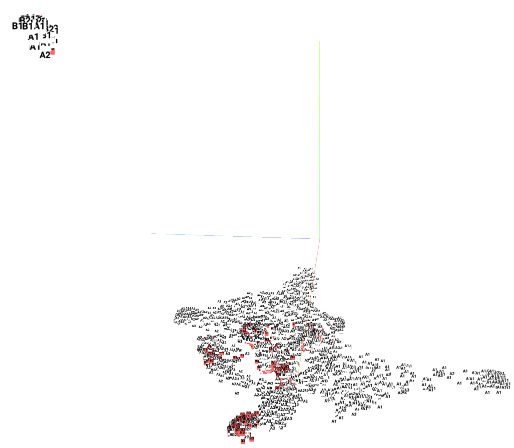
3. Middle stage of GDI-I^3



4. Middle stage of GDI-I^1

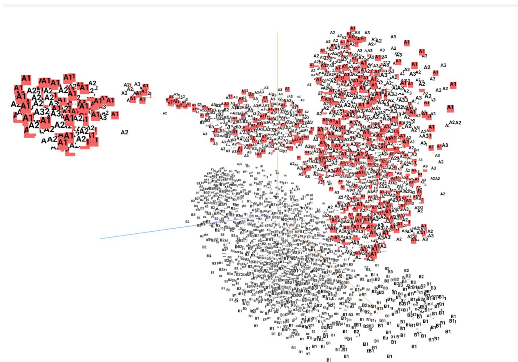


5. Later stage of GDI-I^3

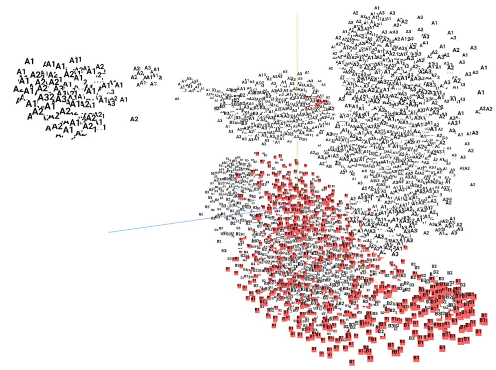


6. Later stage of GDI-I^1

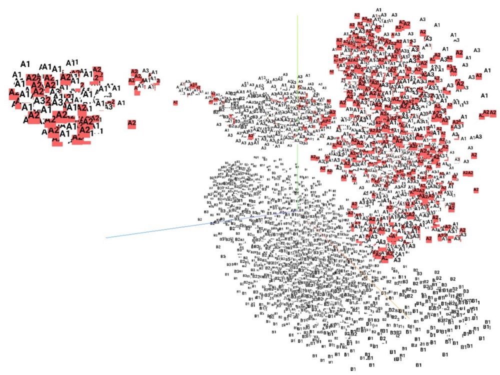
Figure 21. t-SNE of ChopperCommand. t-SNE is drawn from 6k states. We sample 1k states from each stage of GDI-I^3 and GDI-I^1. We highlight 1k states of each stage of GDI-I^3 and GDI-I^1.



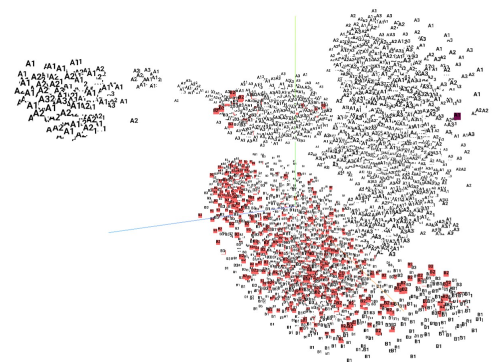
1. Early stage of GDI-H^3



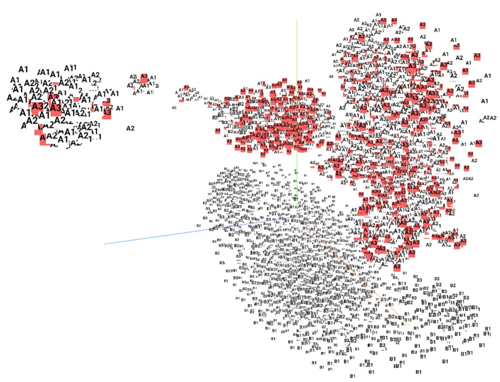
2. Early stage of GDI-I^1



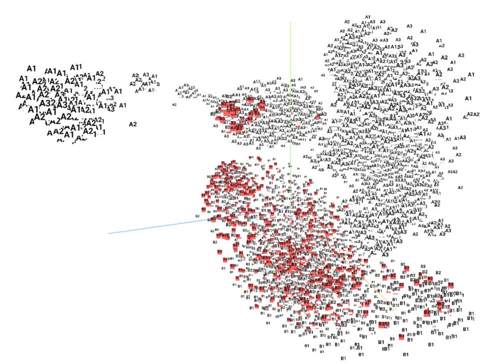
3. Middle stage of GDI-H^3



4. Middle stage of GDI-I^1

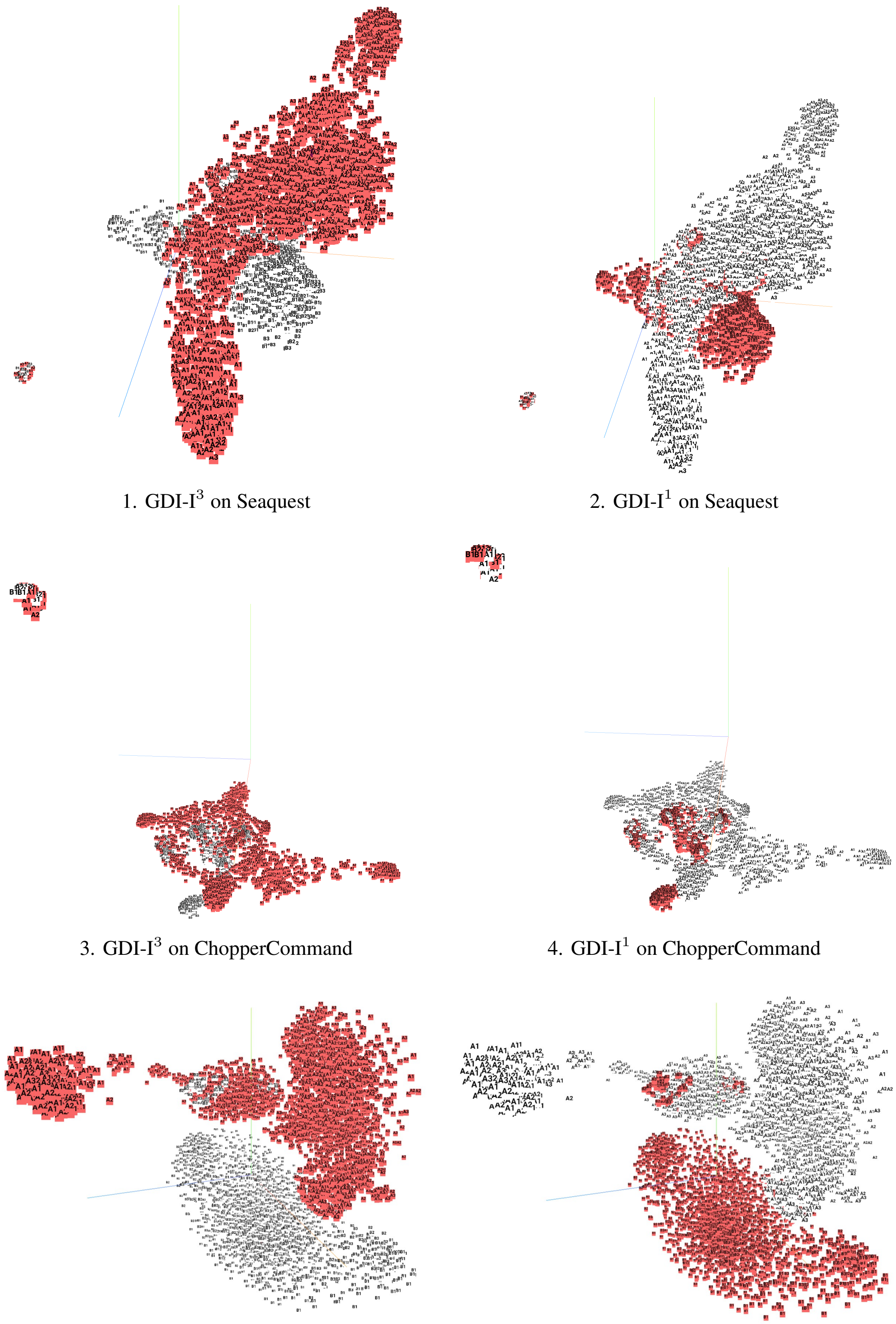


5. Later stage of GDI-H^3



6. Later stage of GDI-I^1

Figure 22. t-SNE of Krull. t-SNE is drawn from 6k states. We sample 1k states from each stage of GDI-H^3 and GDI-I^1. We highlight 1k states of each stage of GDI-H^3 and GDI-I^1.



1. GDI-I^3 on Seaquest

2. GDI-I^1 on Seaquest

3. GDI-I^3 on ChopperCommand

4. GDI-I^1 on ChopperCommand

5. GDI-H^3 on Krull

6. GDI-I^1 on Krull

Figure 23. Overview of t-SNE in Atari games. Each t-SNE figure is drawn from 6k states. We highlight 3k states of GDI-I^3, GDI-H^3 and GDI-I^1, respectively.