Beyond Tabula Rasa: Reincarnating Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning *tabula rasa*, that is without any prior knowledge, is the prevalent workflow in reinforcement learning (RL) research. However, RL systems, when applied to large-scale settings, rarely operate tabula rasa. Such large-scale systems undergo multiple design or algorithmic changes during their development cycle and use *ad hoc* approaches for incorporating these changes without re-training from scratch, which would have been prohibitively expensive. Additionally, the inefficiency of tabula rasa RL typically excludes researchers without access to industrial-scale resources from tackling computationally-demanding problems. To address these issues, we present *reincarnating* RL as an alternative workflow, where prior computational work (*e.g.,* learned policies) is reused or transferred between design iterations of an RL agent, or from one RL agent to another. As a step towards enabling reincarnating RL from any agent to any other agent, we focus on the specific setting of efficiently transferring an existing sub-optimal policy to a standalone value-based RL agent. We find that existing approaches fail in this setting and propose a simple algorithm to address their limitations. Equipped with this algorithm, we demonstrate reincarnating RL's gains over tabula rasa RL on Atari 2600 games, a challenging locomotion task, and the real-world problem of navigating stratospheric balloons. Overall, this work argues for an alternate approach to RL research, which we believe could significantly improve real-world RL adoption and help democratize it further.

1 Introduction

Reinforcement learning (RL) is a general-purpose paradigm for making data-driven decisions. Due to this generality, the prevailing trend in RL research is to learn systems that can operate efficiently *tabula rasa*, that is without much previously learned knowledge about the problem. However, tabula rasa RL systems are typically the exception rather than the norm for solving large-scale RL problems [3, 11, 69, 79]. Such large-scale RL systems often need to function for long periods of time and continually experience new data; restarting them from scratch may require weeks if not months of computation, and there may be billions of data points to re-process – this makes the tabula rasa approach impractical. For example, the system that plays Dota 2 at a superhuman level [11] or the system that manipulates a robot hand for solving Rubik's cube [3], underwent several months of RL training with continual changes (*e.g.,* in model architecture, environment, *etc*) during their development; this necessitated building upon the previously trained system after such changes to circumvent re-training from scratch, which was done using *ad hoc* approaches. Furthermore, tackling challenging problems with deep RL often incurs substantial computational and financial cost: AlphaStar [79], which achieves grandmaster level in StarCraft, was trained using TPUs for more than a month and replicating it would cost several million dollars[1]. As a result, the majority of the RL community outside certain resource-rich labs is currently excluded from tackling such problems.

[1] See Appendix A.1 for an estimate based on publicly available information.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

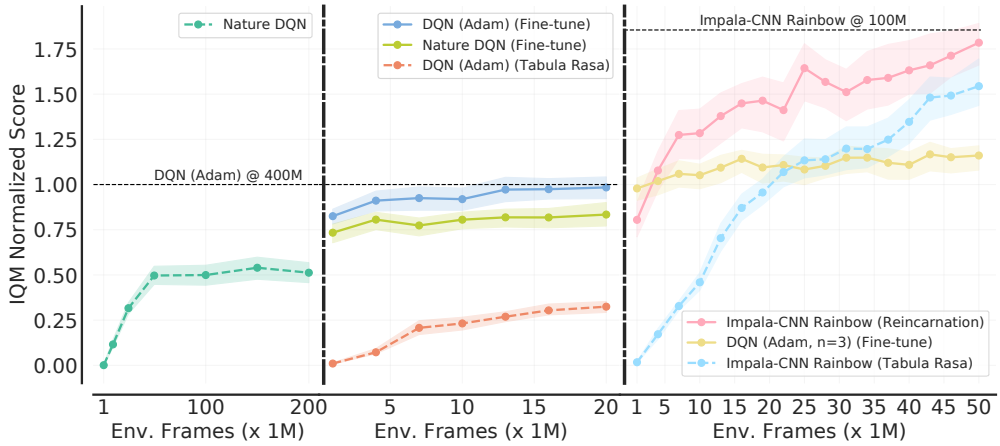

Figure 1: **A reincarnating RL workflow on ALE**. The plots show Interquartile mean (IQM) [2] normalized scores over training, computed using 50 seeds, aggregated across 10 Atari games. The two vertical separators correspond to loading network weights and replay buffer for fine-tuning while offline pre-training on replay buffer using QDagger (Section 4.1) for reincarnation. Shaded regions show 95% confidence intervals. We assign a score of 1 to DQN (Adam) trained for 400M frames and 0 to a random agent. **(Panel 1)** *Tabula rasa* Nature DQN [56] nearly converges in performance after training for 200M environment frames. **(Panel 2)** Reincarnation via fine-tuning Nature DQN with a reduced learning rate leads to 50% higher IQM with only 1M additional frames (leftmost point). Furthermore, fine-tuning Nature DQN while switching from RMSProp [34] to Adam [43] matches the performance of DQN (Adam) trained from scratch for 400M frames, using only 20M frames. **(Panel 3)** A modern ResNet (Impala-CNN [24]) with a better algorithm (Rainbow [32]) outperforms further fine-tuning n-step DQN. Reincarnated Impala-CNN Rainbow outperforms tabula rasa Impala-CNN Rainbow throughout training and requires only 50M frames to nearly match its performance at 100M frames.

To address both the computational and sample inefficiencies of tabula rasa RL, we present *reincarnating* RL as an alternative research workflow. Reincarnating RL focuses on maximally leveraging existing computational work, such as learned network weights and collected data, to accelerate training across design iterations of an RL agent or when moving from one agent to another. In this workflow, agents need not be trained tabula rasa, except for initial forays into new problems. Reincarnating RL can be seen as an attempt to provide a more formal foundation for the prior *ad hoc* strategies to confront the challenges of large-scale RL model development [*e.g.,* 3, 11].

Reincarnating RL also suggests a different benchmarking paradigm that can democratize RL by allowing the broader community to continually improve and update existing trained agents, and even collaboratively tackle problems that are currently infeasible with tabula rasa RL. For example, imagine a researcher who has trained a deep RL agent $\mathcal{A}_1$ for a long time (*e.g.,* weeks), but now this or another researcher wants to experiment with better algorithms or architectures. While the tabula rasa workflow requires re-training another agent from scratch, reincarnating RL provides the more viable option of transferring $\mathcal{A}_1$ to another agent and training this agent further, or simply fine-tuning $\mathcal{A}_1$ (Figure 1). However, beyond some large-scale reincarnation efforts (Section 3), the research community has not focused much on reincarnating RL, in part due to the prevalence of tabula rasa RL. To this end, this work investigates this setting and its potential for accelerating RL research.

As a step towards developing broadly applicable reincarnation approaches, we focus on the specific setting of *policy-to-value* reincarnating RL (PVRL) for efficiently transferring a suboptimal teacher policy to a value-based RL student agent (Section 4). Since it is undesirable to maintain dependency on past teachers for successive reincarnations, we require a PVRL algorithm to "wean" off the teacher dependence as training progresses. We find that prior approaches, when evaluated for PVRL on the Arcade Learning Environment (ALE) [8], either result in small improvements over the tabula rasa student or exhibit degradation when weaning off the teacher. To address these limitations, we introduce QDagger, which combines Dagger [65] with n-step Q-learning, and outperforms prior approaches. Equipped with QDagger, we demonstrate the sample and compute-efficiency gains of reincarnating RL over tabula rasa RL, on ALE and a humanoid locomotion task. Reincarnating RL also makes it easier to make progress on a simulated real-world problem of navigating stratospheric balloons [9] in a computationally feasible manner (Section 5). We also discuss some considerations in reincarnating RL that require further investigation (Section 6). Finally, to improve the benchmarking ecosystem in reincarnating RL, we will open-source our code and trained agents.

2 Preliminaries

The goal in RL is to maximize the long-term discounted reward in an environment. We model the environment as an MDP, defined as $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ [63], with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a stochastic reward function $R(s, a)$, transition dynamics $P(s'|s, a)$ and a discount factor $\gamma \in [0, 1)$. A policy $\pi(\cdot|s)$ maps states to a distribution over actions. The Q-value function $Q^\pi(s, a)$ for a policy $\pi(\cdot|s)$ is the expected sum of discounted rewards obtained by executing action a at state s and following $\pi(\cdot|s)$ thereafter. $Q^\pi(s, a)$ is the fixed point of $Q(s, a) = R(s, a) + \gamma E_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')} [Q(s', a')]$. DQN [56] builds on Q-learning [81] and parameterizes the Q-value function, Q_θ, with a neural net with parameters θ while following an ϵ-greedy policy with respect to Q_θ for data collection. DQN minimizes the temporal difference (TD) loss, $\mathcal{L}_{TD}(\mathcal{D}_S)$, on transition tuples, (s, a, r, s'), sampled from an experience replay buffer $\mathcal{D}_S$ collected during training:

$$\mathcal{L}_{TD}(\mathcal{D}) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(Q_\theta(s, a) - r - \gamma \max_{a'} \bar{Q}_\theta(s', a') \right)^2 \right] \tag{1}$$

where $\bar{Q}_\theta$ is a delayed copy of the same Q-network, referred to as the *target network*. Modern value-based RL agents, such as Rainbow [32], use *n-step* returns to further stabilize learning. Specifically, rather than training the Q-value estimate $Q(s_t, a_t)$ on the basis of the single-step temporal difference error $r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)$, an n-step target $\sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n \max_{a'} Q(s_{t+n}, a') - Q(s_t, a_t)$ is used in the TD loss, with intermediate future rewards stored in the replay $\mathcal{D}$.

3 Related work

Large-scale reincarnation efforts. Several high-profile RL achievements employ a limited form of reincarnating RL. OpenAI Five [11], which can play Dota2 at a superhuman level, required 10 months of large-scale RL training and went through continual changes in code and environment (*e.g.,* expanding observation spaces) during development. To avoid restarting from scratch after such changes, OpenAI Five used "**surgery**" akin to Net2Net [16] style transformations to convert a trained model to certain bigger architectures with custom weight initializations. AlphaStar [79] employs population-based training (**PBT**) [39], which periodically copies weights of the best performing value-based agents and mutates hyperparameters during training. Although PBT and surgery methods are efficient, they have they *can not* be used for reincarnating RL when switching to arbitrary architectures (*e.g.,* feed-forward to recurrent networks) or from one model class to another (*e.g.,* policy to a value function). In this work, we focus on reincarnating RL methods that allow such architecture and algorithm changes. Akkaya et al. [3] trained RL policies for several months to manipulate a robot hand for solving Rubik's cube. To do so, they "rarely trained experiments from scratch" but instead initialized new policies, with architectural changes, from previous trained policies using **behavior cloning** via on-policy distillation [19, 62]. AlphaGo [69] also used behavior cloning on human replays for initializing the policy and fine-tuning it further with RL. However, behavior cloning is only applicable for policy to policy transfer and is inadequate for the PVRL setting of transferring a policy to a value function [*e.g.,* 58, 77]. Several prior works also **fine-tune** existing agents with deep RL for reducing training time, especially on real-world tasks such as chip floor-planning [55], robotic manipulation [40], aligning language models [5], and compiler optimization [76]. In line with these works, we find that fine-tuning a value-based agent can be an effective reincarnation strategy (Figure 7). However, fine-tuning is *constrained* to use the same architecture as the agent being fine-tuned. Instead, we focus on reincarnating RL methods that do not have this limitation.

Leveraging existing agents. Existing policies have been previously used for improving data collection in RL [12, 15, 26, 71, 83]; we evaluate one such recent approach, JSRL [77], which improves exploration in goal-reaching RL tasks. However, our PVRL experiments indicate that JSRL performs poorly on ALE. Closely related to PVRL, Schmitt et al. [68] propose kickstarting to speed-up actor-critic agents using an interactive teacher policy by combining on-policy distillation [19, 62] with RL. Empirically, we find that kickstarting is a strong baseline for PVRL, however it exhibits unstable behavior without n-step returns and underperforms QDagger. PVRL also falls under the framework of agents teaching agents (ATA) [20] with RL-based students and teachers. While ATA approaches, such as action advice [75], emphasize how and when to query the teacher or evaluating the utility of teacher advice, PVRL focuses on sample-efficient transfer and does not impose constraints on querying the teacher. PVRL is also different from prior work on accelerating RL using a heuristic or oracle value function [7, 18, 72], as PVRL only assumes access to a suboptimal policy.

Leveraging prior data. Learning from demonstrations (LfD) [4, 28, 33, 38, 66] approaches focus on accelerating RL training using demonstrations. Such approaches typically assume access to

optimal or near-optimal trajectories, often obtained from human demonstrators, and aim to match the demonstrator's performance. Instead, PVRL focuses on leveraging a suboptimal teacher policy, which can be obtained from any trained RL agent, that we wean off during training. Empirically, we find that DQfD [33], a well-known LfD approach to accelerate deep Q-learning, when applied to PVRL, exhibits severe performance degradation when weaning off the teacher. Rehearsal approaches [57, 61, 70] focus on improving exploration by replaying demonstrations during learning; we find that such approaches are ineffective for leveraging the teacher in PVRL. Offline RL [1, 48, 50] focuses on learning *solely* from fixed datasets while reincarnating RL focuses on leveraging prior information, which can also be presented as offline datasets, for speeding up further learning from environment interactions. Recent work [45, 49, 52, 58] use offline RL to pretrain on prior data and then fine-tune online. We also evaluate this pretraining approach for PVRL. However, PVRL is more flexible than only using a fixed dataset for pretraining, as it assumes access to an interactive teacher policy.

4 Case Study: Policy to Value Reincarnating RL

Reincarnating RL can leverage different ways of representing prior knowledge: logged datasets, learned policies, pretrained models (*e.g.,* value functions, dynamics models), representations, and others. While prior large-scale efforts have used a limited form of reincarnating RL (Section 3), such as transferring one policy to another, it is unclear how to design reincarnation approaches that can be incorporated in any RL research project. To exemplify the challenges of designing such approaches, we focus on *policy-to-value* reincarnating RL (**PVRL**) for transferring a suboptimal teacher policy to a value-based student agent to accelerate learning. While we can obtain a policy from any RL agent, we chose this setting because value-based RL methods (Q-learning, actor-critic) can leverage off-policy data for better sample efficiency. The difficulty in PVRL arises from the fact that policies do not estimate discounted returns but only distribution over actions, while value functions do. To be broadly useful for reincarnating agents, a PVRL algorithm should satisfy the following desiderata:

- **Teacher-agnostic**. Reincarnating RL has limited utility if the student is constrained by the teacher's architecture or learning algorithm. Thus, we require the student to be teacher-agnostic.
- **Weaning**. It is undesirable to maintain dependency on past teachers when reincarnation may occur several times over the course of a project, or one project to another. Thus, it is necessary that the student's dependence on the teacher policy can be weaned off, as training progresses.
- **Sample-efficient**. Naturally, reincarnating RL is only useful if it is computationally cheaper than training from scratch. Thus, it is desirable that the student can recover and possibly improve upon the teacher's performance using fewer environment samples than training tabula rasa.

PVRL on Atari 2600 games. Given the above desiderata for PVRL, we now empirically investigate whether existing methods that leverage existing data or agents (see Section 3) suffice for PVRL. The specific methods that we consider were chosen because they are simple to implement, and also because they have been designed with closely related goals in mind.

Experimental setup. We conduct experiments on ALE with sticky actions [54]. To reduce the computational cost of our experiments, we use a subset of 10 commonly-used Atari 2600 games: Asterix, Breakout, Space Invaders, Seaquest, Q*Bert, Beam Rider, Enduro, Ms Pacman, Bowling and River Raid. We obtain the teacher policy π_T by running DQN [56] with Adam optimizer for 400 million environment frames, requiring 7 days of training per run with Dopamine [14] on P100 GPUs. Without loss of generality, we assume access to a dataset $\mathcal{D}_T$ that can be generated by the teacher. For this work, $\mathcal{D}_T$ corresponds to the final replay buffer (1 million transitions) of the teacher DQN. For a challenging PVRL setting, we use DQN as the student since tabula rasa DQN requires a substantial amount of training to reach the teacher's performance. To emphasize sample-efficient reincarnation, we train this student for only 10 million frames, a 40 *times* smaller sample budget than the teacher. Furthermore, we wean off the teacher at 6 million frames. See Appendix A.4 for more details.

Evaluation. For reliable evaluation, we follow the recommendations of Agarwal et al. [2] and report the Interquartile Mean (IQM) normalized scores with 95% bootstrap confidence intervals (CIs), aggregated across 10 Atari games with 3 seeds each. The normalization is done such that the random policy corresponds to a score of 0 and the teacher policy π_T corresponds to a score of 1. This differs from typically reported human-normalized scores, as we wanted to highlight the performance differences between the student and the teacher. Next, we describe the approaches we investigate.

- **Rehearsal**: Since the student, in principle, can learn using any off-policy data, we can replay teacher data $\mathcal{D}_T$ along with the student's own interactions during training. Following Paine et al.

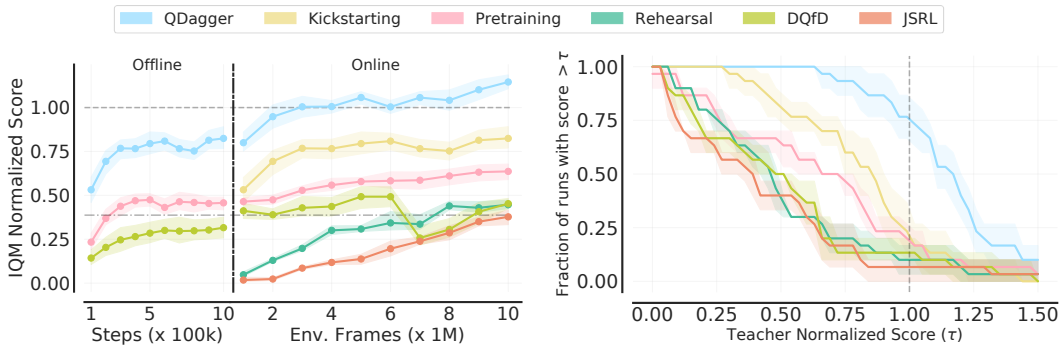

Figure 2: **Comparing PVRL algorithms** for reincarnating a student DQN agent given a teacher policy (with normalized score of 1), obtained from a DQN agent trained for 400M frames (Section 4). Tabula rasa 3-step DQN student ($-\cdot$ line) obtains an IQM teacher normalized score around 0.39. Shaded regions show 95% bootstrap CIs. **Left**. Sample efficiency curves based on IQM normalized scores, aggregated across 10 games and 3 runs, over the course of training. Among all algorithms, only QDagger (Section 4.1) surpasses teacher performance within 10 million frames. **Right**. Performance profiles [2] showing the distribution of normalized scores across all 30 runs at the end of training (higher is better). The area under an algorithm's profile corresponds to its mean performance while τ value where the profile intersects $y = 0.5$ shows its median performance. QDagger stochastically dominates other algorithms and outperforms the teacher in 75% of runs.

[61], the student minimizes the TD loss on mini-batches that contain ρ% of the samples from $\mathcal{D}_T$ and the rest from the student's replay $\mathcal{D}_S$ (different ρ and n-step values in Figure A.11).

- **JSRL** (Figure 3, left): JSRL [77] uses an interactive teacher policy as a "guide" to improve exploration and rolls in with the guide for a random number of environment steps. To evaluate JSRL, we vary the maximum number of roll-in steps, α, that can be taken by the teacher and sample a random number of roll-in steps between $[0, \alpha]$ every episode. As the student improves, we decay the steps taken by the teacher every iteration (1M frames) by a factor of β.
- **RL Pretraining**: Given access to teacher data $\mathcal{D}_T$, we can pre-train the student using offline RL. To do so, we use CQL [46], a widely used offline RL algorithm, which jointly minimizes the TD and behavior cloning on logged transitions in $\mathcal{D}_T$ (Equation A.3). Following pretraining, we fine-tune the learned Q-network using TD loss on the student's replay $\mathcal{D}_S$.
- **Kickstarting** (Figure 3, right): Akin to kickstarting [68], we jointly optimize the TD loss with an on-policy distillation loss on the student's self-collected data in $\mathcal{D}_S$. The distillation loss uses the cross-entropy between teacher's policy π_T and the student policy $\pi(\cdot|s) = \mathrm{softmax}(Q(s, \cdot)/\tau)$, where τ corresponds to temperature. To wean off the teacher, we decay the distillation coefficient as training progresses. Note that kickstarting does not pretrain on teacher data.
- **DQfD** (Figure 4, left): Following DQfD [32], we initially pretrain the student on teacher data D_T using a combination of TD loss with a large margin classification loss to imitate the teacher actions (Equation A.4). After pretraining, we train the student on its replay data $\mathcal{D}_S$, again using a combination of TD and margin loss. While DQfD minimizes the margin loss throughout training, we decay the margin loss coefficient during the online phase, akin to kickstarting.

Results. Rehearsal, with best-performing teacher data ratio ($\rho = 1/16$), is marginally better than tabula rasa DQN but significantly underperforms the teacher (Figure 2, teal), which seems related to the difficulty of standard value-based methods to learn from off-policy teacher data [60]. JSRL does not improve performance compared to tabula rasa DQN and even hurts performance with a large number of teacher roll-in steps (Figure 3, right). The ineffectiveness of JSRL on ALE is likely due to the state-distribution mismatch between the student and the teacher, as the student may never visit the states visited by the teacher and as a result, doesn't learn to correct for its previous mistakes [15].

Pretraining with offline RL on logged teacher data recovers around 50% of the teacher's performance and fine-tuning this pretrained Q-function online marginally improves performance (Figure 2, pink). However, fine-tuning degrades performance with 1-step returns, which is more pronounced with higher values of CQL loss coefficient (Figure A.12). We also find that kickstarting exhibits performance degradation (Figure 3, right), which is severe with 1-step returns, once we wean off the teacher policy. Akin to kickstarting, we again observe a severe performance collapse when weaning off the the teacher dependence in DQfD (Figure 4, left), even when using n-step returns. We hypothesize

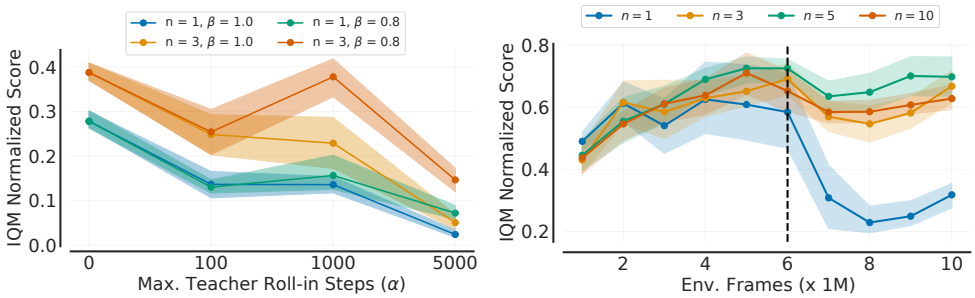

Figure 3: **Left**. **JSRL**. The plot shows teacher normalized scores with 95% CIs, after training for 10M frames, aggregated using IQM across 10 Atari games with 3 seeds each. Each point corresponds to a different experiment, evaluated using 30 seeds, with specific values of JSRL parameters (α, β) and n-step returns. **Right**. **Kickstarting**, with different n-step returns. The plots show IQM scores over the coures of training. Kickstarting exhibits performance degradation, which is severe with 1-step, and is unable to surpass teacher's performance.

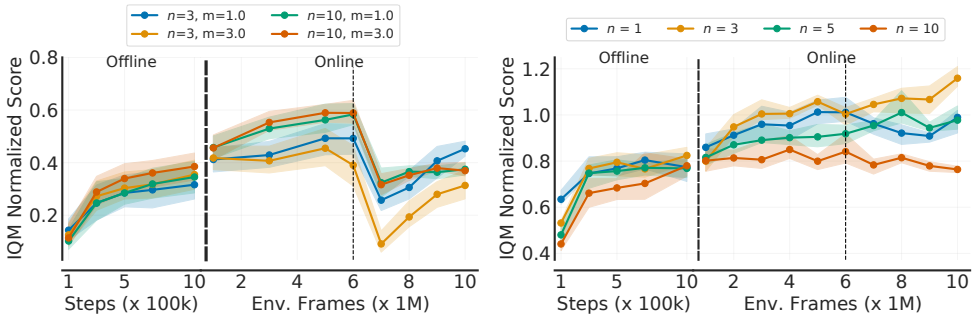

Figure 4: **Left**. **DQfD**. Here, m is the margin loss parameter, which is the loss penalty when the student's action is different from the teacher. **Right**. **QDagger**, with different n-step returns. In both, the 1st vertical line separates pretraining phase from online phase while the 2nd one indicates completely weaning off the teacher.

that this performance degradation is caused by the inconsistency between Q-values trained using a combination of imitation learning and TD losses, as opposed to only minimizing the TD loss. Further investigation of this phenomenon is left for future work. We also find that using intermediate values of n-step returns, such as $n = 3$ (also used by Rainbow [32]), quickly recovers after the performance drop from weaning while larger n-step values impede learning, possibly due to stale target Q-values. See Fedus et al. [25] for a discussion about benefits of n-step returns.

4.1 QDagger: A simple PVRL baseline

To address the limitations of prior approaches, we propose QDagger, a simple method for PVRL that combines Dagger [65], an interactive imitation learning algorithm, with n-step Q-learning (Figure 4, right). Specifically, we first pre-train the student on teacher data $\mathcal{D}_T$ by minimizing $\mathcal{L}_{QDagger}(\mathcal{D}_T)$, which combines distillation loss with the TD loss, weighted by a constant λ. This pretraining phase helps the student to mimic the teacher's state distribution, akin to the behavior cloning phase in Dagger. After pretraining, we minimize $\mathcal{L}_{QDagger}(\mathcal{D}_S)$ on the student's replay $\mathcal{D}_S$, akin to kickstarting, where the teacher "corrects" the mistakes on the states visited by the student. As opposed to minimizing the Dagger loss indefinitely, QDagger decays the distillation loss coefficient λ_t ($\lambda_0 = \lambda$) as training progresses, to satisfy the weaning desiderata for PVRL. Weaning allows QDagger to deviate from the suboptimal teacher policy π_T, as opposed to being perpetually constrained to stay close to π_T (Figure 9). We find that both decaying λ_t linearly over training steps or using an affine function of the ratio of student and teacher performance worked well (Appendix A.4). Assuming the student policy $\pi(\cdot|s) = \mathrm{softmax}(Q(s, \cdot)/\tau)$, the QDagger loss is given by:

$$\mathcal{L}_{QDagger}(\mathcal{D}) = \mathcal{L}_{TD}(\mathcal{D}) + \lambda_t \mathbb{E}_{s\sim\mathcal{D}}\Big[\sum_a \pi_T(a|s) \log \pi(a|s)\Big] \tag{2}$$

Figure 2 shows that QDagger outperforms prior methods and surpasses the teacher. We remark that DQfD can be viewed as a QDagger ablation that uses a margin loss instead of a distillation loss, while kickstarting as another ablation that does not pretrain on teacher data. Equipped with QDagger, we show how to incorporate PVRL into our workflow and demonstrate its benefits over tabula rasa RL.

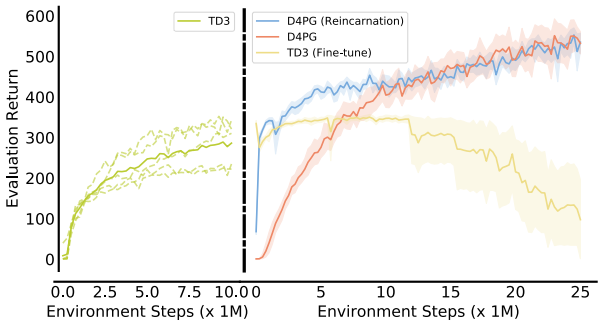

Figure 5: **Reincarnating RL on humanoid:run**. **(Panel 1)**. We observe that TD3 nearly saturates in performance after training for 10M environment steps. The dashed traces show individual runs while the solid line shows the mean return. **(Panel 2)**. Reincarnated D4PG performs better than its tabula rasa counterpart until the first 10M environment steps and then converges to similar performance (with lower variance). Furthermore, training TD3 for a large number of steps eventually results in performance collapse. We use identically parameterized MLP critic and policy networks with 2 hidden layers of size $(256, 256)$ for TD3 and 3 hidden layers of size $(256, 256, 512)$ for D4PG. Shaded regions show 95% CIs based on 10 seeds.

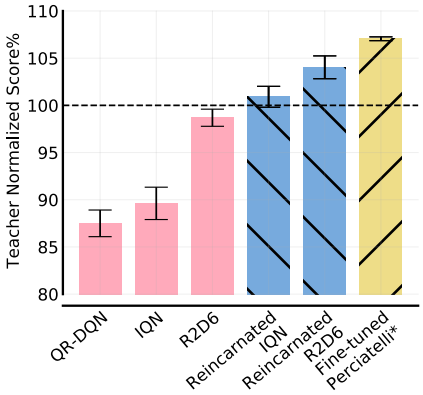

Figure 6: **Comparing BLE agents**. $*$: See main text. We compare QR-DQN [22] with the same MLP architecture as Perciatelli, IQN [21] with DenseNet [37], and R2D6. Reincarnated R2D6 outperforms Perciatelli as well as the tabula rasa agents, but lags behind fine-tuned Perciatelli. We report the mean score (TWR50) across 10,000 evaluation seeds with varying wind difficulty, averaged over 2 independent runs. Error bars show minimum and maximum scores on those runs.

5 Reincarnating RL as a research workflow

Revisiting ALE. As Mnih et al. [56]'s development of Nature DQN established the tabula rasa workflow on ALE, we demonstrate how iterating on ALE agents' design can be significantly accelerated using a reincarnating RL workflow, starting from Nature DQN, in Figure 1. Although Nature DQN used RMSProp, Adam yields better performance than RMSProp [1, 59]. While we can train another DQN agent from scratch with Adam, fine-tuning Nature DQN with Adam, with a reduced learning rate (see Figure 7), matches the performance of this tabula rasa DQN trained for 400M frames, using a 20 *times* smaller sample budget (Panel 2 in Figure 1). As such, on a P100 GPU, fine-tuning only requires training for a few hours rather than a week needed for tabula rasa training. Given this fine-tuned DQN, fine-tuning it further results in diminishing returns with additional frames due to being constrained to use the 3-layer convolutional neural network (CNN) with the DQN algorithm.

Let us now consider how one might use a more general reincarnation approach to improve on fine-tuning, by leveraging architectural and algorithmic advances since DQN, without the sample complexity of training from scratch (Panel 3 in Figure 1). Specifically, using QDagger to transfer the fine-tuned DQN, we reincarnate Impala-CNN Rainbow that combines Dopamine Rainbow [32], which incorporates distributional RL [10], prioritized replay [67] and n-step returns, with an Impala-CNN architecture [24], a deep ResNet with 15 convolutional layers. Tabula rasa Impala-CNN Rainbow outperforms fine-tuning DQN further within 25M frames. Reincarnated Impala-CNN Rainbow quickly outperforms its teacher policy within 5M frames and maintains superior performance over its tabula rasa counterpart throughout training for 50M frames. To catch up with the performance of this reincarnated agent's performance, the tabula rasa Impala-CNN Rainbow requires additional training for 50M frames (48 hours on a P100 GPU). See Appendix A.5 for more training details. Overall, these results indicate how past research on ALE could have been accelerated by incorporating a reincarnating RL approach to designing agents, instead of always re-training agents from scratch.

Tackling a challenging control task. To show how reincarnating RL can enable faster experimentation, we apply PVRL on the *humanoid:run* locomotion task, one of the hardest control problems in DMC [73] due to its large action space (21 degrees of freedom). For this experiment, shown in Figure 5, we use actor-critic agents in Acme [35]. For the teacher policy, we use TD3 [27] trained for 10M environment steps and pick the best run. We find that fine-tuning this TD3 agent degrades performance after 15M environment steps (other learning rates in Appendix A.5), which may be related to capacity loss in value-based RL with prolonged training [47, 53]. For reincarnation, we use single-actor D4PG [6], a distributional RL variant of DDPG [51], with a larger policy and critic architecture than TD3. Reincarnated D4PG performs better than its tabula rasa counterpart for the first 10M environment interactions. Both these agents converge to similar performance, which is

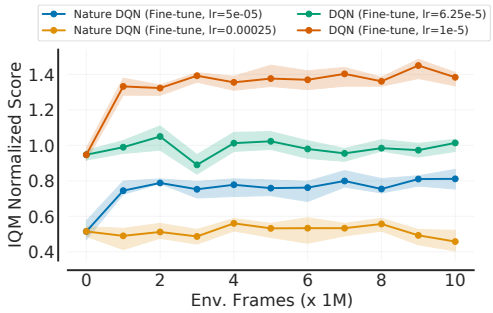

Figure 7: **Reincarnation via fine-tuning** with same and reduced lr, relative to the original agent.

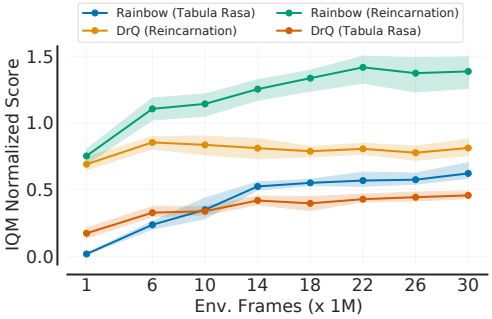

Figure 8: **Contrasting benchmarking** results under tabula rasa and PVRL settings.

likely a limitation of QDagger. This result also raises the question of whether better PVRL methods can lead to reincarnated agents that outperform their tabula rasa counterpart throughout learning. Nevertheless, tabula rasa D4PG requires additional training for 10-12 hours on a V100 GPU to match reincarnated D4PG's performance, which might quickly add up to a substantial savings in compute when running a large set of experiments (*e.g.,* architectural or hyperparameter sweeps).

Balloon Learning Environment (BLE) [30]. One of the motivations for our work is to be able to use deep RL in real-world tasks in a data and computationally efficient manner. To this end, the BLE provides a high-fidelity simulator for navigating stratospheric balloons using RL [9]. An agent in the BLE can choose from three actions to control the balloon: move up, down, or stay in place. The balloon can only move laterally by "surfing" the winds at its altitude; the winds change over time and vary as the balloon changes position and altitude. Thus, the agent is interacting with a *partially observable* and non-stationary system, rendering this environment quite challenging. For the teacher, we use the Perciatelli QR-DQN agent provided by BLE, trained using large-scale distributed RL for 40 days on the *production-level* Loon simulator by Bellemare et al. [9] and further fine-tuned in BLE. For our experiments, we train agents using Acme with 64 distributed actors for a budget of 50,000 episodes on a single cloud TPU-v2 (costs $4.5/hour), taking approximately 10-12 hours per run.

In Figure 6, we compare the final performance of agents trained tabula rasa (in pink), with reincarnation (in blue), and fine-tuned (in yellow). We consider three agents, QR-DQN [22] with an MLP architecture (same as Perciatelli), IQN [21] with a Densenet architecture [37], and a recurrent agent R2D6[2] for addressing the partial observability in BLE. When trained tabula rasa, none of these agents are able to match the teacher performance, with the teacher-lookalike QR-DQN agent performing particularly poorly. As R2D6 and IQN have substantial architectural differences from the teacher, we utilize PVRL for transferring the teacher. Reincarnation allows IQN to match and R2D6 to surpass teacher performance, although both lag behind fine-tuning the teacher.

When fine-tuning, we are reloading the weights from Perciatelli, which was notably trained on a broader geographical region than BLE and whose training distribution can be considered a *superset* of what is used by the other agents; this is likely the reason that fine-tuning does remarkably well relative to other agents in BLE. Efficiently transferring information in Perciatelli's weights to another agent *without* the replay data from the Loon simulator presents an interesting challenge for future work. Overall, the improved efficiency of reincarnating RL (fine-tuning and PVRL) over tabula rasa RL, as evident on the BLE, could make deep RL more accessible to researchers without access to industrial-scale resources as they can build upon prior computational work, such as model checkpoints, enabling the possible reuse of months of prior computation (*e.g.,* Perciatelli).

6 Considerations in Reincarnating RL

Reincarnation via fine-tuning. Given access to model weights and replay of a value-based agent, a simple reincarnation strategy is to fine-tune this agent. While naive fine-tuning with the same learning rate (lr) as the original agent does not exhibit improvement, fine-tuning with a reduced lr, for only 1 million additional frames, results in 25% IQM improvement for DQN (Adam) and 50% IQM improvement for Nature DQN trained with RMSProp (Figure 7). As reincarnating RL leverages existing computational work (*e.g.,* model checkpoints), it allows us to easily experiment with such hy-

[2]R2D6 builds on recurrent replay distributed DQN (R2D2) [41], which uses a LSTM-based policy, and incorporates dueling networks [80], distributional RL [10], DenseNet [37], and double Q-learning [78].

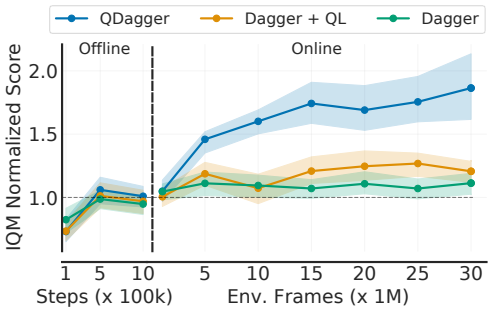

Figure 9: **Reincarnation *vs.* Distillation**. Reincarnating Impala-CNN Rainbow from a DQN (Adam) policy trained for 400M frames, using QDagger, and comparing it to Dagger (imitation learning) and Dagger + Q-learning (imitation-regularized RL).

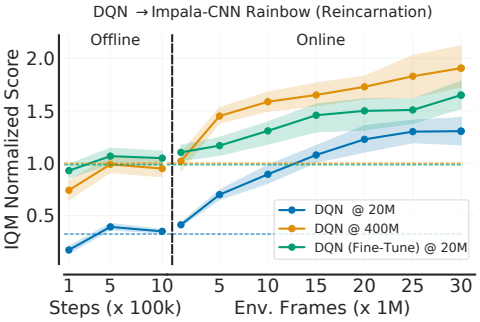

Figure 10: **Reincarnation from different teachers**, namely, a DQN (Adam) policy trained for 20M and 400M frames and fine-tuned Nature DQN in Figure 1 that achieves similar performance to DQN (Adam) trained for 400M frames.

perparameter schedules, which can be expensive in the tabula rasa setting. Note that when fine-tuning, one is *forced* to keep the same network architecture; in contrast, reincarnating RL grants flexibility in architecture and algorithmic choices, which can surpass fine-tuning performance (Figures 1 and 5).

Difference with tabula rasa benchmarking. Are student agents that are more data-efficient when trained from scratch also better for reincarnating RL? In Figure 8, we answer this question in the negative, indicating the possibility of developing better students for utilizing existing knowledge. Specifically, we compare Dopamine Rainbow [32] and DrQ [84], under tabula rasa and PVRL settings. DrQ outperforms Rainbow in low-data regime when trained from scratch but underperforms Rainbow in the PVRL setting as well as when training longer from scratch. Based on this, we speculate that reincarnating RL comparisons might be more consistent with asymptotic tabula rasa comparisons.

Reincarnation *vs.* Distillation. PVRL is different from imitation learning or imitation-regularized RL as it focuses on using an existing policy only as a launchpad for further learning, as opposed to imitating or staying close to it. To contrast these settings, we run two ablations of QDagger for reincarnating Impala-CNN Rainbow given a DQN teacher policy: (1) Dagger [65], which only minimizes the on-policy distillation loss in QDagger, and (2) Dagger + QL, which uses a fixed distillation loss coefficient throughout training (as opposed to QDagger, which decays it; see Equation 2). As shown in Figure 9, Dagger performs similarly to the teacher while Dagger + QL improves over the teacher but quickly saturates in performance. On the contrary, QDagger substantially outperforms these ablations and shows continual improvement with additional environment interactions.

Dependency on prior work. Findings in reincarnating RL are dependent on existing computational work (*e.g.,* teacher policies). This is similar to machine learning areas, such as NLP and computer vision, where building upon pretrained models is the dominant paradigm [*e.g.,* 17, 23, 31, 36]. To investigate teacher dependence in PVRL, we reincarnate a fixed student from three different DQN teachers (Figure 10). As expected, we observe that a higher performing teacher results in a better performing student. However, reincarnation from two distinct policies that perform similarly results in different performance trends. This suggests that a reincarnated student's performance depends not only on the teacher's performance but also on its behavior, which opens up the possibility for creating teacher policies that may be more suited for reincarnation (*e.g.,* pre-trained exploratory policies [13]).

7 Conclusion

This work shows that reincarnating RL is a more compute and data efficient workflow than tabula rasa RL. Nevertheless, our results also open several avenues for future work. Particularly, more research is needed for enabling workflows that can incorporate knowledge provided in a form other than a policy, such as pretrained models or representations, developing better methods for PVRL, and extending PVRL to transfer a policy to model-based agents. We hope that this work motivates RL researchers to release computational work (*e.g.,* model checkpoints) for their publications, which would allow others to directly build on their work. In this regards, we will open-source our code and trained agents. Concurrent to this work, Gogianu et al. [29] released $25,000$ trained Atari agents, which we believe would further facilitate reincarnating RL. As Newton put it "If I have seen further it is by standing on the shoulders of giants", we argue that reincarnating RL can substantially accelerate progress by building on prior computational work, as opposed to always redoing this work from scratch.

Societal Impacts

In this work, we present reincarnating RL an alternative workflow for doing RL research. This workflow could positively impact society by reducing the computational burden on researchers and is more environment friendly than tabula rasa RL. For example, reincarnating RL allow researchers to train super-human Atari agents on a single GPU within a span of few hours as opposed to training for a few days. Additionally, reincarnating RL is more accessible to the wider research community, as researchers without sufficient compute resources can build on prior computational work from resource-rich groups, and even improve upon them using limited resources. Furthermore, this democratization could directly improve RL applicability for practical applications, as most businesses that could benefit from RL often cannot afford the expertise to design in-house solutions. However, this democratization could also make it easier to apply RL for potentially harmful applications. Additionally, reincarnating RL could carry forward the bias or undesirable traits from the previously learned systems. As such, we urge practitioners to be mindful of how RL fits into the wider socio-technical context of its deployment.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[2] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[4] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[5] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[6] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[7] Wissam Bejjani, Rafael Papallas, Matteo Leonetti, and Mehmet R Dogar. Planning with a receding horizon for manipulation in clutter using a learned value function. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE, 2018.

[8] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[9] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

[10] Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2022. http://www.distributional-rl.org.

[11] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[12] Reinaldo AC Bianchi, Carlos HC Ribeiro, and Anna HR Costa. Heuristically accelerated q–learning: a new approach to speed up reinforcement learning. In *Brazilian Symposium on Artificial Intelligence*, pages 245–254. Springer, 2004.

[13] Víctor Campos, Pablo Sprechmann, Steven Hansen, Andre Barreto, Steven Kapturowski, Alex Vitvitskyi, Adria Puigdomenech Badia, and Charles Blundell. Beyond fine-tuning: Transferring behavior in reinforcement learning. *arXiv preprint arXiv:2102.13515*, 2021.

[14] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

[15] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. Learning to search better than your teacher. In *International Conference on Machine Learning*, pages 2058–2066. PMLR, 2015.

[16] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[17] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[18] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[19] Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1331–1340. PMLR, 2019.

[20] Felipe Leno Da Silva, Garrett Warnell, Anna Helena Reali Costa, and Peter Stone. Agents teaching agents: a survey on inter-agent transfer learning. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–17, 2020.

[21] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

[22] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 1, 2018.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[24] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

[25] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

[26] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006.

[27] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[28] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[29] Florin Gogianu, Tudor Berariu, Lucian Buşoniu, and Elena Burceanu. Atari agents, 2022. URL https://github.com/floringogianu/atari-agents.

[30] Joshua Greaves, Salvatore Candido, Vincent Dumoulin, Ross Goroshin, Sameera S. Ponda, Marc G. Bellemare, and Pablo Samuel Castro. Balloon Learning Environment, 12 2021. URL https://github.com/google/balloon-learning-environment.

[31] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[32] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[33] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[34] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[35] Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

[36] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[37] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[38] Peter C Humphreys, David Raposo, Toby Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Alex Goldin, Adam Santoro, et al. A data-driven approach for learning to control computers. *arXiv preprint arXiv:2202.08137*, 2022.

[39] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[40] Ryan Julian, Benjamin Swanson, Gaurav S Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. *arXiv preprint arXiv:2004.10190*, 2020.

[41] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

[42] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[44] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

[45] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[46] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[47] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

[48] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[49] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[50] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[51] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[52] Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, et al. Aw-opt: Learning robotic skills with imitation andreinforcement at scale. In *Conference on Robot Learning*, pages 1078–1088. PMLR, 2022.

[53] Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*, 2022.

[54] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[55] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[56] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[57] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[58] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[59] Johan S Obando-Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning (ICML)*, 2021.

[60] Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[61] Tom Le Paine, Caglar Gulcehre, Bobak Shahriari, Misha Denil, Matt Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil Rabinowitz, Duncan Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

[62] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[63] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[64] Colin Raffel. A call to build models like we build open-source software. https://colinraffel.com/blog/a-call-to-build-models-like-we-build-open-source-software.html, 2021.

[65] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[66] Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[67] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[68] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

[69] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[70] Alexey Skrynnik, Aleksey Staroverov, Ermek Aitygulov, Kirill Aksenov, Vasilii Davydov, and Aleksandr I Panov. Forgetful experience replay in hierarchical reinforcement learning from expert demonstrations. *Knowledge-Based Systems*, 218:106844, 2021.

[71] William D Smart and L Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3404–3410. IEEE, 2002.

[72] Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*, 2018.

[73] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[74] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

[75] Lisa Torrey and Matthew Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060, 2013.

[76] Mircea Trofin, Yundi Qian, Eugene Brevdo, Zinan Lin, Krzysztof Choromanski, and David Li. Mlgo: a machine learning guided compiler optimizations framework. *arXiv preprint arXiv:2101.04808*, 2021.

[77] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*, 2022.

[78] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, number 1, 2016.

[79] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[80] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[81] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[82] Lilian Weng. Meta reinforcement learning. *lilianweng.github.io*, 2019. URL https://lilianweng.github.io/posts/2019-06-23-meta-rl/.

[83] Linhai Xie, Sen Wang, Stefano Rosa, Andrew Markham, and Niki Trigoni. Learning with training wheels: speeding up training with a simple controller for deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6276–6283. IEEE, 2018.

[84] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 5 (BLE and continuous control results) and Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix A.2
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.4 and A.5
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] 95% CIs
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.3

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Apache License, Version 2.0
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]