Reincarnating Reinforcement Learning: Reusing Prior Computation to Accelerate Progress

Rishabh Agarwal[1,2*] **Max Schwarzer**[1,2]
Pablo Samuel Castro[1] **Aaron Courville**[2] **Marc G. Bellemare**[1,2]
[1] Google Research, Brain Team [2] MILA

Abstract

Learning *tabula rasa*, that is without any previously learned 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 deep 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 or class of problem settings, 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 alternative approach to RL research, which we believe could significantly improve real-world RL adoption and help democratize it further. Open-sourced code and trained agents at `agarwl.github.io/reincarnating_rl`.

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 learned knowledge including prior computational work such as offline datasets or learned policies. However, tabula rasa RL systems are typically the exception rather than the norm for solving large-scale RL problems [4, 13, 55, 75, 85]. 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 human-like level [13] underwent several months of RL training with continual changes (*e.g.,* in model architecture, environment, *etc*) during its 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 (described in Section 3).

Current RL research also excludes the majority of researchers outside certain resource-rich labs from tackling complex problems, as doing so often incurs substantial computational and financial cost: AlphaStar [85], which achieves grandmaster level in StarCraft, was trained using TPUs for more than a month and replicating it would cost several million dollars (Appendix A.1). Even the quintessential deep RL benchmark of training an agent on 50+ Atari games [10], with at least 5 runs, requires

---

[*] Correspondence to Rishabh Agarwal <`rishabhagarwal@google.com`>.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

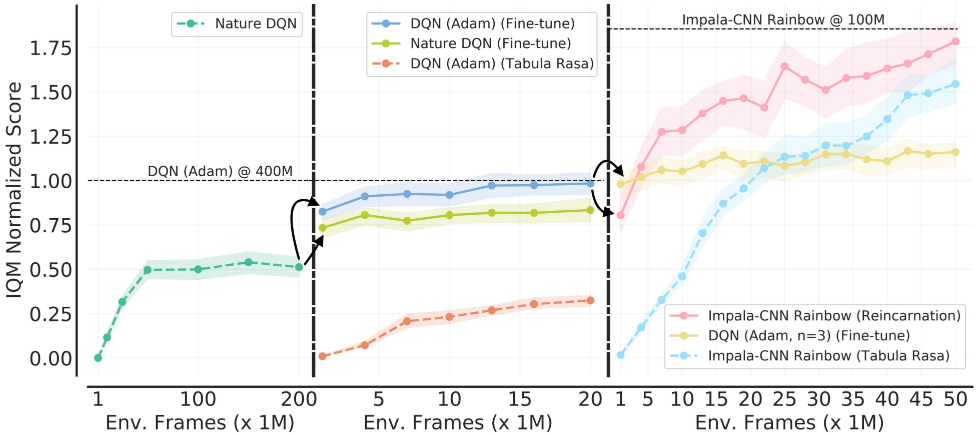

Figure 1: **A reincarnating RL workflow on ALE**. The plots show IQM [2] normalized scores over training, computed using 50 seeds, aggregated across 10 Atari games. The vertical separators correspond to loading network weights and replay buffer for fine-tuning while offline pre-training on replay buffer 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 [60] nearly converges in performance after training for 200M 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 to Adam matches the performance of DQN (Adam) trained from scratch for 400M frames, using only 20M frames. **(Panel 3)**. A modern ResNet (Impala-CNN [26]) with a better algorithm (Rainbow [35]) outperforms further fine-tuning n-step DQN. **Reincarnating Impala-CNN Rainbow from DQN**, outperforms tabula rasa Impala-CNN Rainbow throughout training and requires only 50M frames to nearly match its performance at 100M frames. See Section 5.

more than 1000 GPU days. As deep RL research move towards more challenging problems, the computational barrier to entry in RL research is likely to further increase.

To address both the computational and sample inefficiencies of tabula rasa RL, we present *reincarnating* RL (RRL) as an alternative research workflow or a class of problems to focus on. RRL seeks to *maximally leverage 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 RRL, agents need not be trained tabula rasa, except for initial forays into new problems. For example, imagine a researcher who has trained an agent $\mathcal{A}_1$ for a long time (*e.g.,* weeks), but now this or another researcher wants to experiment with better architectures or RL algorithms. 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). As such, RRL can be viewed as an attempt to provide a formal foundation for the research workflow needed for real-world and large-scale RL models.

Reincarnating RL can democratize research by allowing the broader community to tackle larger-scale and complex RL problems without requiring excessive computational resources. As a consequence, RRL can also help avoid the risk of researchers overfitting to conclusions from small-scale RL problems. Furthermore, RRL can enable a benchmarking paradigm where researchers continually improve and update existing trained agents, especially on problems where improving performance has real-world impact (*e.g.,* balloon navigation [11], chip design [59], tokamak control [24]). Furthermore, a common real-world RL use case will likely be in scenarios where prior computational work is available (*e.g.,* existing deployed RL policies), making RRL important to study. However, beyond some *ad hoc* large-scale reincarnation efforts (Section 3), the community has not focused much on studying reincarnating RL as a research problem in its own right. To this end, this work argues for developing general-purpose RRL approaches as opposed to *ad hoc* solutions.

Different RRL problems can be instantiated depending on how the prior computational work is provided: logged datasets, learned policies, pretrained models, representations, *etc*. 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) [10], 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 [71] 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, a humanoid locomotion task and the simulated real-world problem of navigating stratospheric balloons [11] (Section 5). Finally, we discuss some considerations in RRL as well as address reproducibility and generalizability concerns.

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 $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ [69], 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. DQN [60] builds on Q-learning [87] 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 [35], 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

Prior *ad hoc* reincarnation efforts. While several high-profile RL achievements have used reincarnation, it has typically been done in an ad-hoc way and has limited applicability. OpenAI Five [13], which can play Dota 2 at a human-like 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 [17] style transformations to convert a trained model to certain bigger architectures with custom weight initializations. AlphaStar [85] employs population-based training (**PBT**) [42], 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). Akkaya et al. [4] 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 [20, 67]. AlphaGo [75] 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.,* 63, 83]. Contrary to such approaches, we apply reincarnation in settings where these approaches are not applicable including transferring a DQN agent to Impala-CNN Rainbow in ALE, and a distributed agent with MLP architecture to a recurrent agent in BLE. 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 [59], robotic manipulation [43], aligning language models [6], and compiler optimization [82]. 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 often *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 prior computation. While areas such as offline RL, imitation learning, transfer in RL, continual RL *etc* focus on developing methods to leverage prior computation, such areas don't strive to change how we do RL research by incorporating such methods as a part of our workflow. For completeness, we contrast closely related approaches to PVRL, the RRL setting we study.

- **Leveraging existing agents**. Existing policies have been previously used for improving data collection [14, 16, 28, 77, 89]; we evaluate one such approach, JSRL [83], which improves exploration in goal-reaching RL tasks. However, our PVRL experiments indicate that JSRL performs poorly on ALE. Schmitt et al. [74] propose kickstarting to speed-up actor-critic agents using an interactive teacher policy by combining on-policy distillation [20, 67] 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) [21] with RL-based students and teachers. While ATA approaches, such as action advice [81], 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 [9, 19, 78], as PVRL only assumes access to a suboptimal policy. Unlike PVRL methods that wean off the teacher, imitation-regularized RL methods [51, 61] stay close to the suboptimal teacher, which can limit the student's performance with continued training (Figure 9).

- **Leveraging prior data**. Learning from demonstrations (LfD) [5, 30, 36, 40, 72] 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 [36], 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 [62, 66, 76] 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, 49, 52] 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, 51, 55, 63] use offline RL to pretrain on prior data and then fine-tune online. We also evaluate this pretraining approach for PVRL and find that it underperforms QDagger, which utilizes the interactive teacher policy in addition to the prior teacher collected data.

4 Case Study: Policy to Value Reincarnating RL

While prior large-scale efforts have used a limited form of reincarnating RL (Section 3), it is unclear how to design more broadly applicable RRL approaches. To exemplify the challenges of designing such approaches, we focus on the RRL setting for accelerating training of a student agent given access to a suboptimal teacher policy and some data from it. While a policy-based student can be easily reincarnated in this setting via behavior cloning [*e.g.,* 4], we study the more challenging *policy-to-value* reincarnating RL (**PVRL**) setting for transferring a policy to a value-based student agent. 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. 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.
- **Compute & sample efficient**. Naturally, RRL 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 [57]. 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 [60] with Adam optimizer for 400 million environment frames, requiring 7 days of training per run with Dopamine [15] on P100 GPUs.

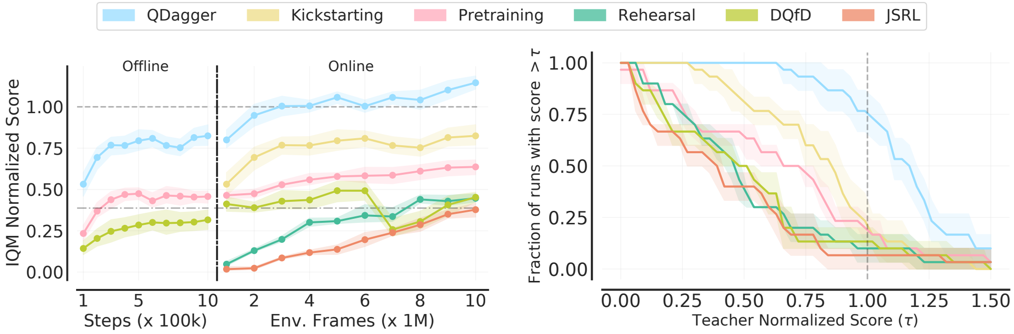

Figure 2: **Comparing PVRL algorithms** for reincarnating a student DQN agent given a teacher policy (with normalized score of 1), obtained from a DQN agent trained for 400M frames (Section 4). Baselines include kickstarting [74], JSRL [83], rehearsal [66], offline pretraining [46] and DQfD [36]. Tabula rasa 3-step DQN student (—· line) obtains an IQM teacher normalized score around 0.39. Shaded regions show 95% 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 scores across all 30 runs at the end of training (higher is better). Area under an algorithm's profile is its mean performance while τ value where it intersects $y = 0.5$ shows its median performance. QDagger outperforms the teacher in 75% of runs.

We also assume access to a dataset $\mathcal{D}_T$ that can be generated by the teacher (see Appendix A.5 for results about dependence on $\mathcal{D}_T$). For this work, $\mathcal{D}_T$ is the final replay buffer (1M transitions) logged by the teacher DQN, which is 100 *times* smaller than the data the teacher was trained on. 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.3 for more details.

Evaluation. Following Agarwal et al. [2], we report interquartile mean normalized scores with 95% confidence intervals (CIs), aggregated across 10 games with 3 seeds each. The normalization is done such that the random policy obtains a score of 0 and the teacher policy π_T obtains 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. [66], 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.12).
- **JSRL** (Figure 3, left): JSRL [83] 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.
- **Offline 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 [74], 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 $\pi(\cdot|s) = \text{softmax}(Q(s, \cdot)/\tau)$, 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 [35], 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.

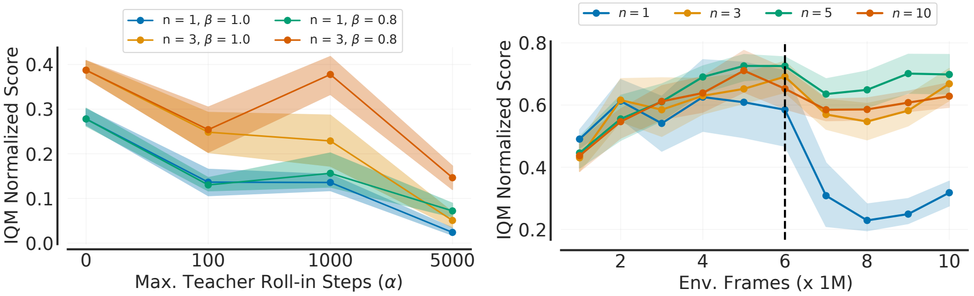

Figure 3: **Left**. **JSRL**. The plot shows teacher normalized scores with 95% CIs, after training for 10M frames, aggregated using IQM across 10 Atari games with 3 seeds each. Each point corresponds to a different experiment, evaluated using 30 seeds, with specific values of JSRL parameters (α, β) 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.

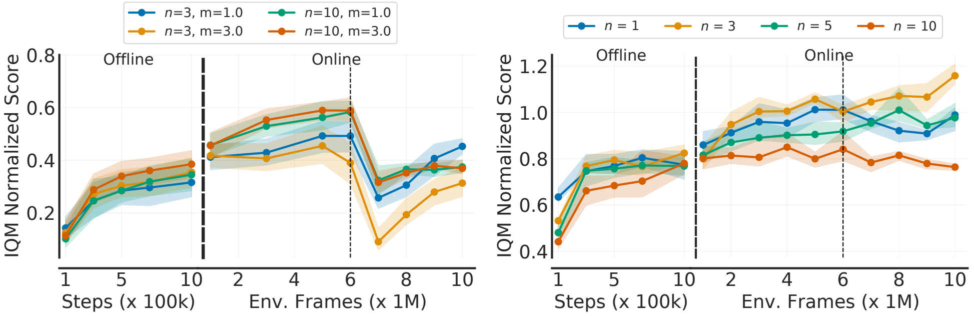

Figure 4: **Left**. **DQfD**. Here, m is the margin loss parameter, which is the loss penalty when the student's action is different from the teacher. **Right**. **QDagger**, with different n-step returns. In both, the 1^{st} vertical line separates pretraining phase from online phase while the 2^{nd}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

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 [65]. 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, left). 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 [16].

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.13). 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 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. We also find that using intermediate values of n-step returns, such as $n = 3$ (also used by Rainbow [35]), quickly recovers after the performance drop from weaning while larger n-step values impede learning, possibly due to stale target Q-values. These results reveal the sensitivity of prior methods in the PVRL setting to specific hyperparameter choices (n-step), indicating the need for developing stable PVRL methods that do not fail when weaning off the teacher. For practitioners, the takeaway is to consider this hyperparameter sensitivity when weaning off the teacher for reincarnation.

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 [71], 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.3). 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.

5 Reincarnating RL as a research workflow

Revisiting ALE. As Mnih et al. [60]'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, 64]. While we can train another DQN agent from scratch with Adam, fine-tuning Nature DQN with Adam and 3-step returns, with a reduced learning rate (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 RL. 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 [35], which incorporates distributional RL [12], prioritized replay [73] and n-step returns, with an Impala-CNN architecture [26], 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.4 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 [80] due to its large action space (21 degrees of freedom). For this experiment, shown in Figure 5, we use actor-critic agents in Acme [37]. For the teacher policy, we use TD3 [29] 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.4), which may be related to capacity loss in value-based RL with prolonged training [47, 56]. For reincarnation, we use single-actor D4PG [8], a distributional RL variant of DDPG [54], 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 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) [33]. One of the motivations of 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

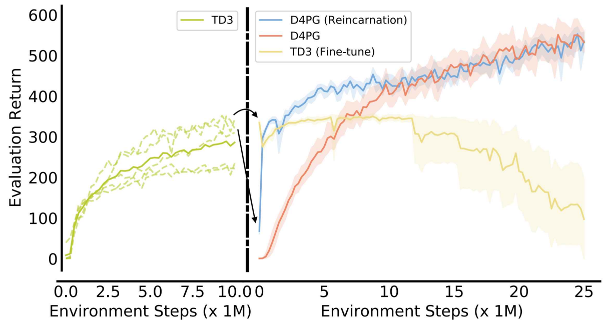

Figure 5: **Reincarnating RL on humanoid:run**. **(Panel 1)**. We observe that TD3 nearly saturates in performance after training for 10M environment steps. The dashed traces show individual runs while the solid line shows the mean return. **(Panel 2)**. Reincarnated D4PG performs better than its tabula 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 but larger networks with 3 hidden layers for D4PG. Shaded regions show 95% CIs based on 10 seeds.

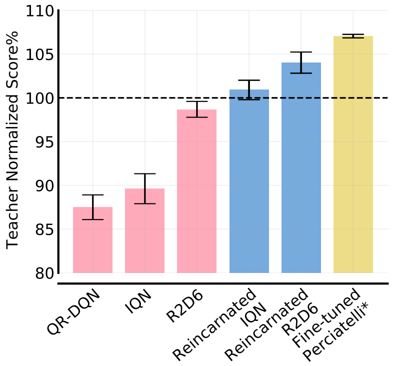

Figure 6: **Comparing BLE agents**. $*$: See main text. We compare QR-DQN [23] with the same MLP architecture as Perciatelli, IQN [22] with DenseNet [39], and R2D6. Reincarnated R2D6 outperforms Perciatelli as well as the tabula rasa agents, but lags behind fine-tuned Perciatelli. We report the mean 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.

provides a high-fidelity simulator for navigating stratospheric balloons using RL [11]. An agent in 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 QR-DQN agent provided by BLE, called *Perciatelli*, trained using large-scale distributed RL for 40 days on the *production-level* Loon simulator by Bellemare et al. [11] and further fine-tuned in BLE. For our experiments, we train distributed RL agents using Acme with 64 actors for a budget of 50,000 episodes on a single cloud TPU-v2, taking approximately 10-12 hours per run.

In Figure 6, we compare the final performance of distributed agents trained tabula rasa (in pink), with reincarnation (in blue), and fine-tuned (in yellow). We consider three agents, QR-DQN [23] with an MLP architecture (same as Perciatelli), IQN [22] with a Densenet architecture [39], 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, although both lag behind fine-tuning the teacher. More details in Appendix A.4.2.

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 nearly saturated original agent does not exhibit improvement, fine-tuning

---

[2] R2D6 builds on recurrent replay distributed DQN (R2D2) [44], which uses a LSTM-based policy, and incorporates dueling networks [86], distributional RL [12], DenseNet [39], and double Q-learning [84].

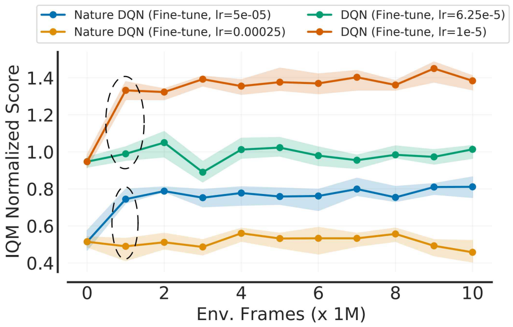

Figure 7: **Reincarnation via fine-tuning** with same and reduced lr, relative to the original agent.

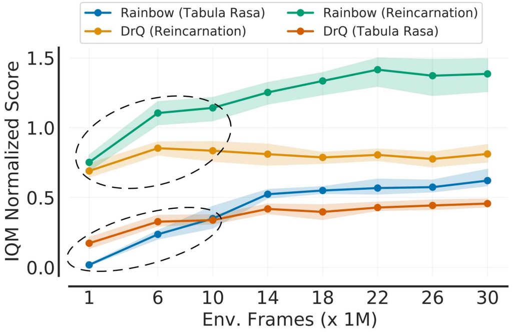

Figure 8: **Contrasting benchmarking** results under tabula rasa and PVRL settings.

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 hyperparameter 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 [35] and DrQ [90], 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 [71], 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. While performance in reincarnating RL depend on prior computational work (*e.g.,* teacher policy in PVRL), this is analogous to how fine-tuning results in NLP / computer vision depend on the pretrained models (*e.g.,* using BERT vs GPT-3). 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 policies with similar performance but obtained from different agents, DQN (Adam) *vs.* a fine-tuned Nature DQN, results in different performance. This suggests that a reincarnated student's performance depends not only on the teacher's performance but also on its behavior. Nevertheless, the ranking of PVRL algorithms remains consistent across these two teacher policies (Figure A.11). See Section 7 for a broader discussion about generalizability.

7 Reproducibility, Comparisons and Generalizability in Reincarnating RL

Scientific Comparisons. Fairly comparing reincarnation approaches entails using the exactly same computational work and workflow. For example, in the PVRL setting, the same teacher and data should be used when comparing different algorithms, as we do in Section 4. To enable this, it would be beneficial if the researchers can release model checkpoints and the data generated (at least the final replay buffers), in addition to open-source code for their trained RL agents. Indeed, to allow others to use the same reincarnation setup as our work, we have already open-sourced DQN (Adam) agent checkpoints and the final replay buffer at `gs://rl_checkpoints`.

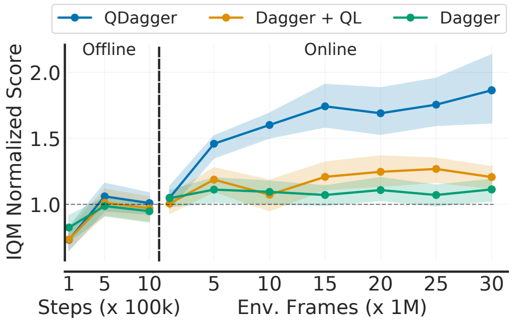

Figure 9: **Reincarnation *vs.* Distillation**. Reincarnating Impala-CNN Rainbow from a DQN (Adam) trained for 400M frames, using QDagger, and comparing it to Dagger (imitation) and Dagger + Q-learning (imitation-regularized RL).

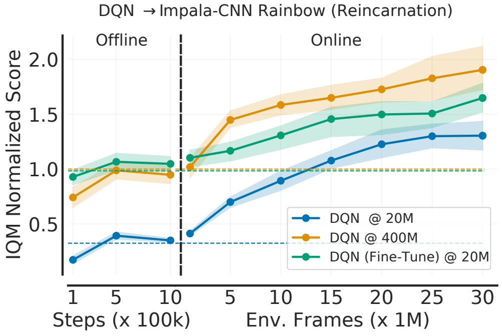

Figure 10: **Reincarnation from different teachers**, namely, a DQN (Adam) policy trained for 20M and 400M frames and fine-tuned Nature DQN in Figure 1 that achieves similar performance to DQN (Adam) trained for 400M frames.

Generalizability. The generalizable findings in reincarnating RL would be about comparing algorithmic efficacy given access to existing computational work on a task. As such, the performance ranking of reincarnation algorithms is likely to remain consistent across different teachers. In fact, we empirically verified this for the PVRL setting, where we find that while using two different teacher policies, namely DQN(Adam) *vs.* a fine-tuned Nature DQN, leads to different performance trends but the ranking of PVRL algorithms remain consistent: QDagger > Kickstarting > Pretraining (see Figure 2 and Figure A.11). Practitioners can use the findings from reincarnating RL to try to improve on an existing deployed RL policy (as opposed to being restricted to running tabula rasa RL). For example, this work developed QDagger using ALE and applied it to PVRL on other tasks with existing policies (Humanoid-run and BLE).

Reproducibility. Reproducibility *from scratch* is challenging in RRL as it would require details of the generation of the prior computational work (*e.g.,* teacher policies), which may itself has been obtained via reincarnating RL. As reproducibility from scratch involves reproducing existing computational work, it could be more expensive than training tabula rasa, which beats the purpose of doing reincarnation. Furthermore, reproducibility from scratch is also difficult in NLP and computed vision, where existing pretrained models (*e.g.,*, GPT-3) are rarely, if ever, reproduced / re-trained from scratch but almost always used as-is. Despite this difficulty, *pretraining-and-fine-tuning* is a dominant paradigm in NLP and vision [*e.g.,* 18, 25, 34, 38], and we believe that a similar difficulty in RRL should not prevent researchers from investigating and studying this important class of problems. Instead, we expect that RRL research would build on **open-sourced** prior computational work. Akin to NLP and vision, where typically a small set of pretrained models are used in research, we believe that research on developing better reincarnating RL methods can also possibly converge to a small set of open-sourced models / data on a given benchmark, *e.g.,* the agents and data we released on Atari or the $25,000$ trained Atari agents released by Gogianu et al. [31], concurrent to this work.

8 Conclusion

Our work shows that reincarnating RL is a much computationally efficient research workflow than tabula rasa RL and can help further democratize research. Nevertheless, our results also open several avenues for future work. Particularly, more research is needed for developing better PVRL methods, and extending PVRL to learn from multiple suboptimal teachers [48, 53], and enabling workflows that can incorporate knowledge provided in a form other than a policy, such as pretrained models [41, 79], representations [88], skills [50, 58, 68], or LLMs [3]. Furthermore, we believe that reincarnating RL would be crucial for building embodied agents in open-ended domains [7, 27, 32]. Aligned with this work, there have been calls for collaboratively building and continually improving large pre-trained models in NLP and vision [70]. We hope that this work motivates RL researchers to release computational work (*e.g.,* model checkpoints), which would allow others to directly build on their work. In this regards, we have open-sourced our code and trained agents with their final replay. Furthermore, re-purposing existing benchmarks, akin to how we use ALE in this work, can serve as testbeds for 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

Reincarnating RL 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. Furthermore, 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.

Acknowledgments

We would like to thank David Ha, Evgenii Nikishin, Karol Hausman, Bobak Shahriari, Richard Song, Alex Irpan, Andrey Kurenkov for their valuable feedback on this work. We thank Joshua Greaves for helping us set up RL agents for BLE. We also acknowledge Ted Xiao, Dale Schuurmans, Aleksandra Faust, George Tucker, Rebecca Roelofs, Eugene Brevdo, Pierluca D'Oro, Nathan Rahn, Adrien Ali Taiga, Bogdan Mazoure, Jacob Buckman, Georg Ostrovski and Aviral Kumar for useful discussions.

Author Contributions

Rishabh Agarwal led the project from start-to-finish, defined the scope of the work to focus on policy to value reincarnation, came up with a successful algorithm for PVRL, and performed the literature survey. He designed, implemented and ran most of the experiments on ALE, Humanoid-run and BLE, and wrote the paper.

Max Schwarzer helped run DQfD experiments on ALE and as well as setting up some agents for the BLE codebase with Acme, was involved in project discussions and edited the paper. Work done as a student researcher at Google.

Pablo Samuel Castro was involved in project discussions, helped in setting up the BLE environment and implemented the initial Acme agents, and helped with paper editing.

Aaron Courville advised the project, helped with project direction and provided feedback on writing.

Marc Bellemare advised the project, challenged Rishabh to come up with an experimental paradigm in which one continuously improves on an existing agent, and provided feedback on writing.

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] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[4] 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.

[5] 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.

[6] 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.

[7] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

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

[24] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[25] 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.

[26] 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.

[27] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *arXiv preprint arXiv:2206.08853*, 2022.

[28] 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.

[29] 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.

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

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

[32] Djordje Grbic, Rasmus Berg Palm, Elias Najarro, Claire Glanois, and Sebastian Risi. Evocraft: A new challenge for open-endedness. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 325–340. Springer, 2021.

[33] 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`.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] Andrew Hundt, Aditya Murali, Priyanka Hubli, Ran Liu, Nakul Gopalan, Matthew Gombolay, and Gregory D. Hager. "good robot! now watch this!": Repurposing reinforcement learning for task-to-task transfer. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=Pxs5XwId51n`.

[42] 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.

[43] 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.

[44] 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.

[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] Andrey Kurenkov, Ajay Mandlekar, Roberto Martin-Martin, Silvio Savarese, and Animesh Garg. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. *arXiv preprint arXiv:1909.04121*, 2019.

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

[50] Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.

[51] 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.

[52] 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.

[53] Siyuan Li, Fangda Gu, Guangxiang Zhu, and Chongjie Zhang. Context-aware policy reuse. *arXiv preprint arXiv:1806.03793*, 2018.

[54] 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.

[55] 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.

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

[57] 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.

[58] Michael Matthews, Mikayel Samvelyan, Jack Parker-Holder, Edward Grefenstette, and Tim Rocktäschel. Hierarchical kickstarting for skill transfer in reinforcement learning. *arXiv preprint arXiv:2207.11584*, 2022.

[59] 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.

[60] 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.

[61] Ted Moskovitz, Michael Arbel, Jack Parker-Holder, and Aldo Pacchiano. Towards an understanding of default policies in multitask policy optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 10661–10686. PMLR, 2022.

[62] 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.

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

[64] 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.

[65] 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.

[66] 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.

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

[68] Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020.

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

[70] 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.

[71] 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.

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

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

[74] 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.

[75] 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.

[76] 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.

[77] 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.

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

[79] Yanchao Sun, Ruijie Zheng, Xiyao Wang, Andrew E Cohen, and Furong Huang. Transfer RL across observation feature spaces via model-based regularization. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=7KdAoOsI81C`.

[80] 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.

[81] 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.

[82] 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.

[83] 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.

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

[85] 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.

[86] 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.

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

[88] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[89] 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.

[90] 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 reproducibility and evaluation concerns in Sec 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] `agarwl.github.io/reincarnating_rl`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.3 and A.4
   (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.2
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]

A Appendix

A.1 AlphaStar cost estimation

We estimate the cost of AlphaStar [85] based on the following description in the paper: "In StarCraft, each player chooses one of three races — Terran, Protoss or Zerg — each with distinct mechanics. We trained the league using three main agents (one for each StarCraft race), three main exploiter agents (one for each race), and six league exploiter agents (two for each race). Each agent was trained using 32 third-generation tensor processing units (TPUv3) over 44 days."

This corresponds to a total of 12 agents (= 3 main + 3 exploiter + 6 league) trained for a total of 1056 TPU hours = 44 days * 24 hours/day on 32 TPU v3 chips. As per current pricing at `https://cloud.google.com/tpu/pricing`, TPUv3 (v3-8) cost \$8 per hour. Based on this, we estimate the cost of replicating AlphaStar results for an independent researcher would be at $1056 * 12 * 32 * \$8 = \$3,244,032$. Please note that we are only considering the cost of replication and not accounting for other costs such as hyperparameter tuning and evaluation.

A.2 Compute resources for PVRL experiments

For experiments in section 4, we used a P100 GPU. For obtaining the teacher policy, the cost of running the tabula rasa DQN for 400M frames for 10 games on 3 seeds each roughly amounts to 7 days x 30 = 210 days of compute on a P100 GPU. For each of the PVRL methods, we trained on 10 games with 3 runs each for 10M frames, which roughly translates to 4-5 hours. For offline pretraining, we train methods for 1 million gradient steps with a batch size of 32, which roughly amounts to 6-7 hours (this could be further sped up by using large-batch sizes). We list the number of configurations we evaluated for each method below.

- Rehearsal: We tried 5 values of teacher data ratio $\rho \times 4$ n-step values, amounting to a total of 20 configurations.
- JSRL: We tried 4 values of teacher roll-in steps $\alpha \times 2$ values of decay parameter $\beta \times 2$ values of n-step, amounting to 16 configurations.
- RL Pretraining: We tried 2 values of $\lambda \times 4$ values of n-step, amounting to 8 configurations.
- Kickstarting: We report results for 4 values of n-step for a specific temperature and distillation loss coefficient. For hyperparameter tuning, we evaluated 2 temperature values and 2 loss coefficients with a specific n-step. Overall, this corresponds to a total of 8 configurations.
- DQfD: We report results for 4 values of n-step $\times$ 2 values of margin loss coefficients $\times$ 2 values of margin parameters, amounting to 16 configurations.
- QDagger: We report results for 4 n-step values for a specific temperature and distillation loss coefficient. For hyperparameter tuning, we evaluated 2 different temperature and loss coefficients with a specific n-step, akin to kickstarting. Overall, this amounts to 8 configurations.

Based on the above, we evaluated a total of 60 (=20 + 16 + 16 + 8) configurations without pretraining while 32 configurations with pretraining. Each of these configurations was evaluated for 30 seeds. This amounts to a total compute time of 300-375 days for runs without pretraining on offline data while 400-480 days of GPU compute for runs involving pretraining, resulting in a total compute time of around 700 - 855 days on a P100 GPU.

A.3 PVRL: Experimental details

Atari 2600 Games: The subset of games in the paper includes games from the original Atari training set used by Bellemare et al. [10] (Asterix, Beam Rider, Seaquest and Space Invaders) as well as validation games used by Mnih et al. [60] (Breakout, Enduro, River Raid), except the games which are nearly solved by DQN, such as Freeway and Pong. We do not use any hard exploration games as a DQN teacher does not provide a meaningful teacher policy for such games. The remaining three games were chosen to test the student agent in environments with challenging characteristics such as requiring planning as opposed to being reactive (*e.g.,* Ms Pacman, Q*Bert), and sparse-reward games that require long-term predictions (Bowling). Furthermore, most of these games can be significantly improved over the teacher DQN performance, which is sub-human on half of the games (Ms Pacman, Q*Bert, Bowling, Seaquest and Beam Rider). Refer to Table A.2 for per-game teacher scores.

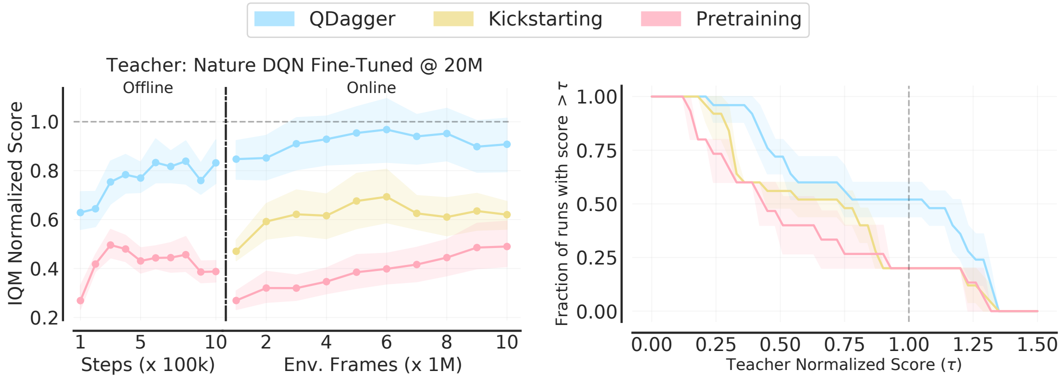

Figure A.11: **Comparison of best-performing PVRL algorithms** for reincarnating a student DQN agent given a teacher policy and replay buffer from a Nature DQN agent trained for 200M frames followed by fine-tuning with Adam for 20M frames (Panel 2 in Figure 1). While the performance of the reincarnated agents depends on the teacher, the ranking of PVRL algorithms remain consistent wrt Figure 2, that is, QDagger > Kickstarting > offline RL pretraining. Shaded regions show 95% bootstrap CIs. **Left**. Sample efficiency curves based on IQM normalized scores, aggregated across 10 games and 3 runs. **Right**. Performance profiles.

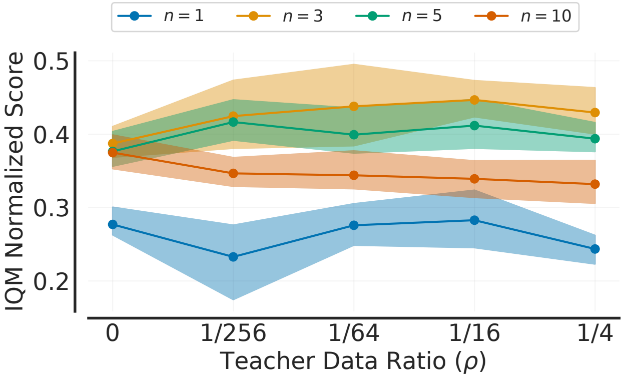

Figure A.12: **Rehearsal** for PVRL. The plots show IQM teacher normalized scores after training for 10M frames, aggregated across 10 Atari games with 3 seeds each. Each point in the above plots correspond to a distinct experiment setting evaluated using 30 seeds. Shaded regions show 95% CIs [2].

Common hyperparameters. We list the hyperparameters shared by all PVRL methods in Table A.1. For all methods, we swept over n-step returns in $\{1, 3, 5, 10\}$ except for DQfD [36], which originally used 10-step returns, where we only tried 3-step and 10-step returns and JSRL for which we tried only 1-step and 3-step returns. For PVRL methods without any pretraining phase, we use a learning rate of $6.25e^{-5}$ (JSRL, Rehearsal), following the hyperparameter configuration for Dopamine [15]. For methods that pretrain on offline data, we sweep over $\{6.25e^{-5}, 1e^{-5}\}$ and found the learning rate of $1e^{-5}$ to perform better in our early experimentation and use it for our main results. We discuss the method specific hyperparameters below. For obtaining the teacher policy using value-based agent, we use the $\mathrm{softmax}(Q_T(s, \cdot)/\tau)$ over the teacher's Q-function Q_T and use the same temperature coefficient τ for both the student and teacher policy.

- **Rehearsal**: We tried 5 different values of the teacher data ratio (ρ) in $\{0, 1/256, 1/64, 1/16, 1/4\}$. The loss, $\mathcal{L}_{Rehearsal}$, can be written as: $\mathcal{L}_{Rehearsal} = \rho\mathcal{L}_{TD}(\mathcal{D}_T) + (1-\rho)\mathcal{L}_{TD}(\mathcal{D}_S)$. As can be seen from the results in Figure A.12, we find a small value of teacher data ratio ($\rho = 1/16$) with 3-step returns to be the best performing configuration.

- **JSRL**. As shown in Figure 4 (left), we swept over the maximum number of teacher roll in steps of α in $\{0, 100, 1000, 5000\}$, and the decay parameter β, which governs how fast we decay the roll-in steps, in $\{0.8, 1.0\}$. Note that $\beta = 1.0$ corresponds to JSRL-Random, which was found to be competitive in performance to JSRL [83].

Table A.1: **Common hyperparameters** used by PVRL experiments using a DQN student agent on ALE in Section 4. These hyperparameters are based on the ones used by the Jax implementation of DQN in Dopamine [15].

Hyperparameter	Setting
Sticky actions	Yes
Sticky action probability	0.25
Grey-scaling	True
Observation down-sampling	(84, 84)
Frames stacked	4
Frame skip (Action repetitions)	4
Reward clipping	[-1, 1]
Terminal condition	Game Over
Max frames per episode	108K
Discount factor	0.99
Mini-batch size	32
Target network update period	every 2000 updates
Min replay history	20000 steps
Environment steps per training iteration	250K
Update period every	4 environment steps
Training ϵ	0.01
Training ϵ-decay steps	50K
Evaluation ϵ	0.001
Evaluation steps per iteration	125K
Q-network: channels	32, 64, 64
Q-network: filter size	8×8, 4×4, 3×3
Q-network: stride	4, 2, 1
Q-network: hidden units	512
Hardware	P100 GPU
Offline gradient steps per iteration	100K
Offline training iterations	10
Offline learning rate	0.0001

- **RL Pretraining**: We use CQL, which optimizes the following loss:

$$\mathcal{L}_{Pretrain} = \mathcal{L}_{TD}(\mathcal{D}_T) + \lambda \mathbb{E}_{s,a\sim\mathcal{D}_T}\left[\log\Big(\sum_{a'} Q(s,a')\Big) - Q(s,a)\right] \tag{A.3}$$

  The choice of CQL is motivated by its simplicity as well as recent findings that offline RL methods that do not estimate the behavior policy are more suited for online fine-tuning [63].

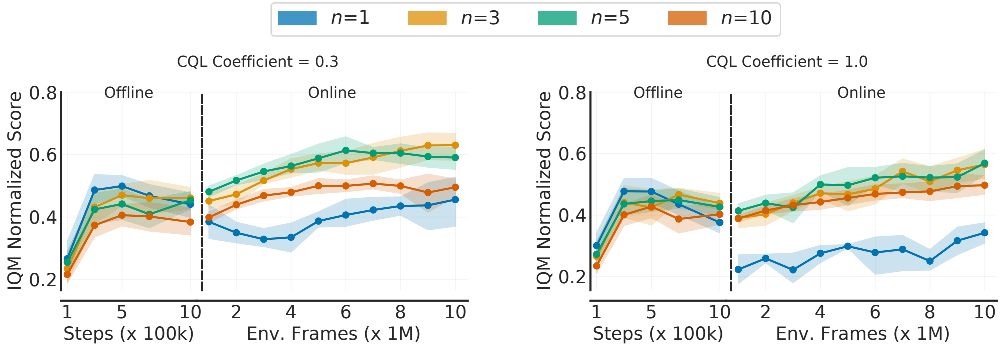

Figure A.13: **RL pretraining** using CQL [46], followed by fine-tuning with **Left**. CQL coefficient 0.3, and **Right**. CQL coefficient 1.0. The plots shows IQM teacher normalized scores over the course of training, computed across 30 seeds, aggregated across 10 Atari games. Online fine-tuning degrades performance with 1-step returns, which is more pronounced with higher CQL loss coefficient.

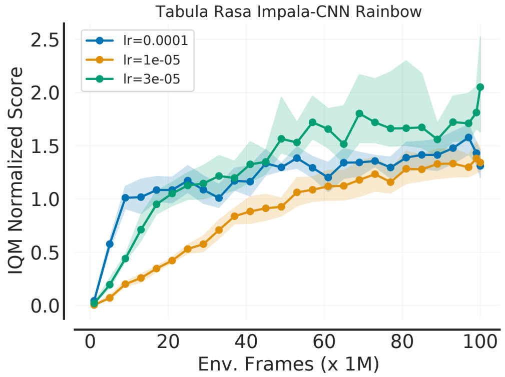

Figure A.14: **Tabula rasa Impala-CNN Rainbow**. Results for different learning rates for the tabula rasa Impala-CNN Rainbow agent. Learning rate of $3e-5$ performs the best. The plots show IQM normalized scores aggregated scores 10 Atari games with 3 seeds, while the shaded regions show 95% bootstrap CIs.

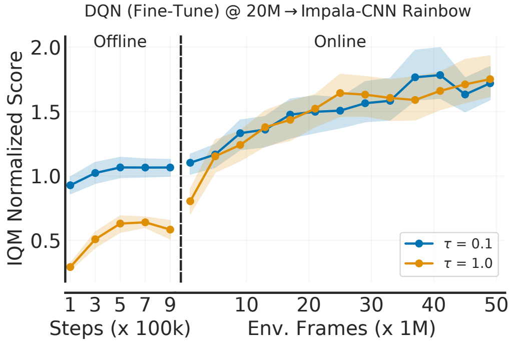

Figure A.15: **Effect of QDagger temperature** τ on the performance of reincarnated Impala CNN-Rainbow in Figure 1. A lower temperature coefficient (0.1) results in better performance in the offline pretraining phase but converges to similar performance in the online phase.

We tried two different values of CQL coefficient λ (0.3 and 1.0), as shown in Figure A.13, and report the results for the better performing coefficient ($\lambda = 0.3$) in the main paper.

- **Kickstarting**. Kickstarting [74] uses the same loss as QDagger (Equation 2), but as discussed in the main paper, kickstarting does not have any pretraining phase on offline data. For the temperature hyperparameter τ for obtaining the policy $\pi(\cdot|s) = \mathrm{softmax}(Q(s,\cdot)/\tau)$, we tried 0.1 and 1.0. Similarly, we swept over two initial values for distillation loss coefficient (λ_0), namely 1.0 and 3.0. For experiments with the DQN student, we found the coefficient 3.0 and temperature 0.1 to perform the best.

- **DQfD**. Following Hester et al. [36], DQfD uses the following loss for training:

  $$\mathcal{L}_{DQfD}(\mathcal{D}) = \mathcal{L}_{TD}(\mathcal{D}) + \eta_t \mathbb{E}_{s\sim\mathcal{D}}\left[\max_a \left(Q(s,a) + f(a_T(s), a)\right) - Q(s, a_T(s))\right] \quad \text{(A.4)}$$

  where η_t corresponds to the margin loss coefficient, $a_T(s) = \mathrm{argmax}_a \pi_T(a|s)$ and $f(a_T, a)$ is a margin function that is 0 when $a = a_T$ and a positive margin m otherwise. We swept over the values $\{1.0, 3.0\}$ for both the margin parameter m and initial values η_0 of the margin loss coefficient η_t. For the DQN student, we report the results for the better performing margin coefficients in the main paper.

- **QDagger**. Akin to kickstarting, we swept over the values of temperature τ in $\{0.1, 1.0\}$ and distillation coefficient λ_0 in $\{1.0, 3.0\}$. For ALE experiments, we decay the distillation loss coefficient every training iteration (1M environment frames) using the fraction of expected returns obtained by the student policy π compared to the teacher policy π_T, that is, $\lambda_t = \mathbf{1}_{t<t_0} \max(1 - G^{\pi}/G^{\pi}_T, 0)$. For fair comparisons with other methods, we use the same strategy for decaying the distillation loss coefficient λ_t for Kickstarting and the margin loss coefficient η_t for DQfD.

A.4 Reincarnating RL as a workflow: Additional details

Revisting ALE. We used the final model checkpoints of Nature DQN [60] from Agarwal et al. [1], which was trained for 200M frames using the hyperparameters in Dopamine [15]. The fine-tuned DQN (Adam) in panel 2 in Figure 1 uses 3-step returns. For the tabula rasa Impala-CNN Rainbow, we use similar hyperparameters to Dopamine Rainbow except for learning rate (lr), for which we ran a sweep over $\{1e-4, 1e-5, 3e-5\}$, shown in Figure A.14, and use the best performing lr of $3e-5$. For the reincarnated Impala-CNN Rainbow in Panel 3, we use a QDagger distillation coefficient of 1.0 and sweep over temperature parameter τ in $\{0.1, 1.0\}$, as shown in Figure A.15. Consistent with our other fine-tuning results on ALE, in Panel 3, using a reduced lr of $3e-6$ for fine-tuning the already fine-tuned DQN agent results in better performance, compared to using an lr of $1e-5$, as used by fine-tuned DQN.

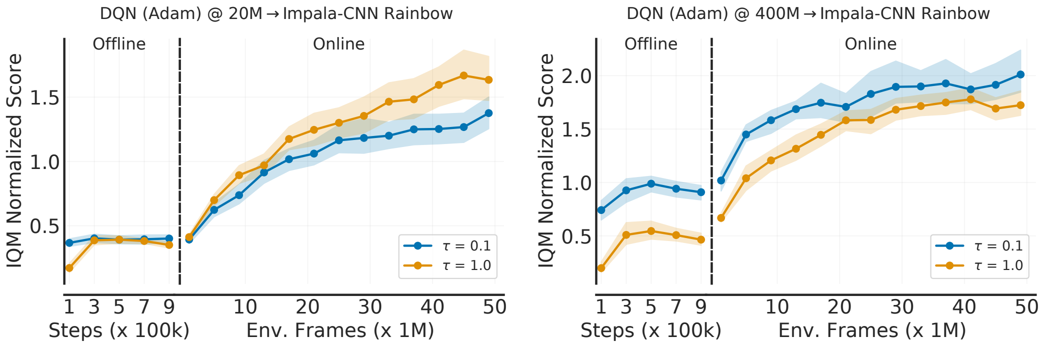

Figure A.16: **Effect of QDagger temperature** τ on the performance of reincarnating Impala CNN-Rainbow from **Left**. DQN (Adam) @ 20M, and **Right**. DQN (Adam) @ 400M. Notably, the better performing temperature τ is dependent on the teacher policy. Lower τ results in cloning a more "spikier" teacher policy. With a reasonably good teacher policy (DQN @ 400M), τ value of 0.1 performs better than 1.0 while with a more suboptimal teacher policy (DQN @ 20M), the higher temperature coefficient of 1.0 performs better than 0.1.

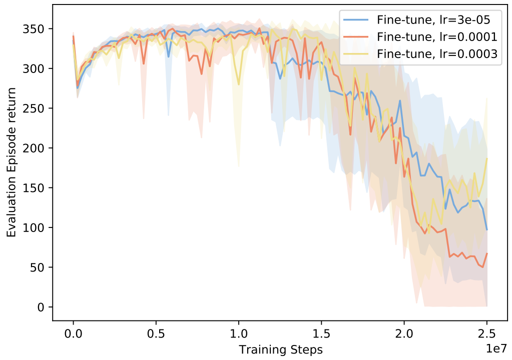

Figure A.17: **Fine-tuning TD3**. Results for fine-tuning a trained Acme TD3 agent with different learning rates. All the different learning rates exhibit similar performance trends including severe degradation after prolonged training.

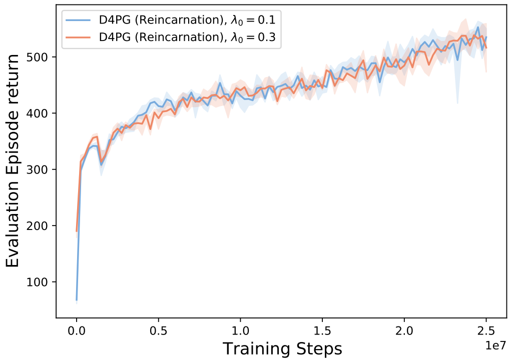

Figure A.18: **Effect of varying initial QDagger distillation coefficient** λ_0 on the performance of reincarnated D4PG. Higher coefficient (0.3) results in faster transfer but converges to similar performance to the runs with lower coefficient (0.1).

A.4.1 Humanoid:Run

The purpose of humanoid:run experiments is to show the utility of reincarnation in a complex continuous control environment given access to a pretrained policy and some data from its final replay buffer. Irrespective of performance from fine-tuning performance TD3, we demonstrate the benefits of using reincarnation over tabula rasa D4PG. For example, reincarnated D4PG achieves performance obtained by SAC in 10M frames[3] in only half the number of samples (*i.e.,* 5M frames).

For obtaining the pretrained policy, TD3 was chosen as (1) it's a well-known method for continuous control, and (2) we could use an off-the-shelf JAX implementation in Acme, which is competitive on humanoid-run. For tabula rasa TD3 [29], we used a learning rate (lr) of $3e-4$ for both the policy and critic, which are represented using a MLP with 2 hidden layers of size (256, 256). For other hyperparameters, we used the default values in Acme's TD3 implementation.

For fine-tuning TD3, we use the last 500K environment steps from the TD3's replay buffer and use the same hyperparameters as tabula rasa TD3 except for lr. Specifically, we swept over the lr in $\{1e-4, 3e-4, 3e-05\}$, shown in Figure A.17, and find that all lrs exhibit performance degradation with prolonged training. We hypothesize that this degradation is likely caused by network capacity loss with prolonged training in value-based RL methods [47, 56]. We believe that this issue does not affect our conclusions about the efficacy of reincarnating RL over a tabula rasa workflow. Further investigation of this performance degradation is outside the scope of the present work.

---

[3] See `https://github.com/denisyarats/pytorch_sac` for SAC learning curves.

For the tabula rasa D4PG [8], we use MLP networks with 3 hidden layers of size $(256, 256, 256)$ for the policy and $(512, 512, 256)$ for the critic. We used $3e-4$ as the learning rate and a sigma value of 0.2 that sets the variance of the Gaussian noise to the behavior policy. Other hyperparameters use the default values for D4PG implementation in Acme. For reincarnating D4PG using QDagger, we minimize a distillation loss between the D4PG's actor policy and the teacher policy from TD3 jointly with the actor-critic losses. For QDagger specific hyperparameters, we pretrain using 200K gradient updates as well as decay λ_t to 0 over a period of 200K gradient updates during the online training phase. Additionally, we sweep over distillation coefficient λ_0 in $\{0.1, 0.3\}$, as shown in Figure A.18. Note that both TD3 and D4PG use 5-step returns by default in Acme.

A.4.2 Balloon Learning Environment (BLE)

Details about BLE can be found in [33]. For self-containedness, we include important details below.

Station Keeping. Stratospheric balloons are filled with a buoyant gas that allows them to float for weeks or months at a time in the stratosphere, about twice as high as a passenger plane's cruising altitude. Though there are many potential variations of stratospheric balloons, the kind emulated in the BLE are equipped with solar panels and batteries, which allow them to adjust their altitude by controlling the weight of air in their ballast using an electric pump. However, they have no means to propel themselves laterally, which means that they are subject to wind patterns in the air around them. By changing its altitude, a stratospheric balloon can surf winds moving in different directions.

BLE Evaluation. The goal of a BLE agent is to station-keep, *i.e.,* to control a balloon to stay within 50 km of a fixed ground station by changing its altitude to catch winds that it finds favorable. We measure how successful an agent is at station-keeping by measuring the fraction of time the balloon is within the specified radius, denoted TWR50 (*i.e.,* the time within a radius of 50km).

Environment Details. The observation space in BLE is a 1099 dimensional array of continuous and boolean values. An agent in 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 More details can be found at `https://balloon-learning-environment.readthedocs.io/en/latest/environment.html`.

Perciatelli Teacher and Data. Perciatelli is the name of the RL agent (QR-DQN with a MLP architecture) trained by Bellemare et al. [11] using a distributed RL setup for more than a month in the production-level Loon simulator and released with the BLE environment after fine-tuning for 14M steps in BLE. For the data, we use the final replay buffer (of size 2M transitions) logged by the Perciatelli agent during BLE fine-tuning.

Architectures. The network architecture used by QR-DQN [23] and Perciatelli is a multilayer perceptron (MLP) with 7 hidden layers of size 600 each with ReLU activations and approximate the distribution using 51 fixed quantiles. For the DenseNet architecture [39] employed by IQN [22] and R2D6, we use 7 hidden layers of size 512 each (for TPU-efficiency), which contains significantly more parameters than the Perciatelli MLP. Additionally, R2D6 uses a LSTM layer of size 512 on top of the DenseNet encoder with QR-DQN loss, while IQN samples 128 quantiles for minimizing the implicit quantile regression loss.

Hyperparameters. We set most hyperparameters for our distributed RL agents based on configuration of the BLE Quantile agent, which can be found at agents/configs/quantile.gin. For tabula rasa agents, we swept over the lr in $\{2e-6, 6e-6, 1e-5\}$ and found $1e-5$ to be the best performing lr. However, for fine-tuning Perciatelli, we found that a lower lr of $1e-6$ performs better. For reincarnated R2D6 and IQN, we use a lr of $1e-5$ and $6e-6$ respectively during the online phase while $2 \times lr$ in the offline pretraining phase. We set distillation temperature τ to be equal to 1.0 and linearly decay the QDagger distillation coefficient λ_t over a fixed number of learner steps (1M for R2D6 and 160000 for IQN). Furthermore, we swept over the initial value of λ_t (λ_0) in $\{0.3, 1.0\}$ and found 1.0 to be better for R2D6 while 0.3 for IQN.

A.5 Additional ablations for QDagger

Dependence on offline replay data. In the PVRL setting on ALE, we assumed access to the last replay buffer (containing 1M transitions) of the teacher DQN (Adam) agent trained for 400M frames. This assumption is aligned with maximally reusing existing computational work and often holds

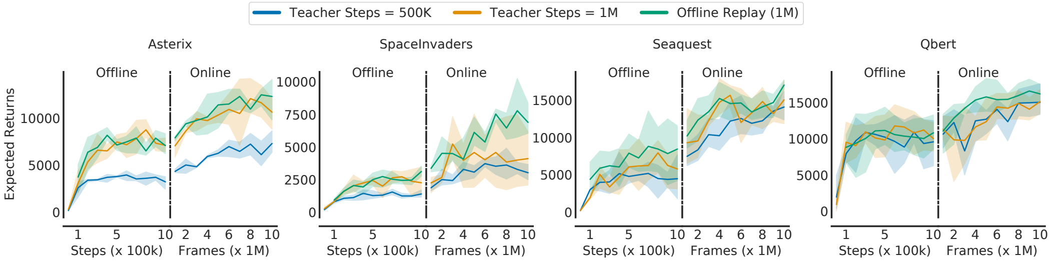

Figure A.19: **Substituting offline replay with teacher collected data** for pretraining phase of QDagger for reincarnating a student DQN given access to the teacher policy obtained from a DQN (Adam) trained for 400M frames. For collecting data using the teacher, we roll out the teacher policy with ϵ-greedy exploration where we decay ϵ from 1 to 0 linearly for the number of teacher steps. Note that 1 transition corresponds to 4 frames in Atari (due to an action repeat of 4).

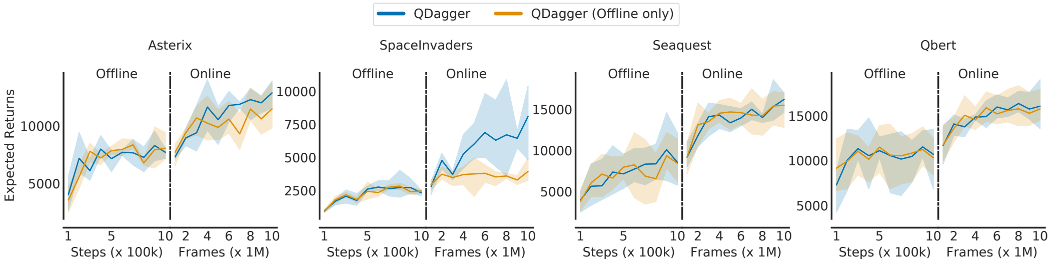

Figure A.20: **Effect of online phase of QDagger** on the performance of a DQN student reincarnated using QDagger, given access to the teacher policy and final replay buffer obtained from a DQN (Adam) trained for 400M frames. While using QDagger only during offline phase performs comparably to QDagger during both online and offline phase on Seaquest and Q*Bert, the online phase leads to slight improvement on Asterix and significant improvements on Space Invaders.

in practice as last replay buffer of RL agents is typically logged on disk in addition to their policy checkpoints. However, it is possible that we only have access to the teacher policy. In this scenario, even if we don't have access to the teacher's replay buffer, we can generate data using the teacher policy during the online RL phase (this would cost environment samples) and use this teacher collected data for pretraining with QDagger loss (analogous to the behavior cloning phase in Dagger).

To verify this empirically, we ran a QDagger ablation on 4 games where we do not assume access to the teacher's replay buffer. As shown in Figure A.19, we found that collecting the same amount of teacher data as the offline replay leads to comparable performance on 3/4 games (Asterix, Qbert, Seaquest). Somewhat surprisingly, collecting only 500K transitions from teacher results in comparable performance to using 1M transitions, except on Asterix where we see substantially lower performance. The slightly better performance with offline replay could be related to its better diversity than teacher collected data. While we used ϵ-greedy exploration with the teacher policy, the offline replay data is collected using possibly diverse policies, as the teacher agent was continually updated during training.

Effect of QDagger loss in online phase. In this work, we use QDagger loss during both the offline pretraining and online RL phase (Equation 2). To evaluate the benefit of QDagger loss in the online phase on ALE, we ran an ablation on 4 games where we use standard Q-learning instead of QDagger loss during the online RL phase. As shown in Figure A.20, we see that the online phase helps improve performance on some games (*e.g.,* Space Invaders, Asterix) while not having much effect on others. The impact of the QDagger loss in the online phase also depends on the offline replay buffer size. For example, on the more complex BLE domain, we only had access to a small amount of replay data relative to the training budget of the Perciatelli teacher. With this teacher, using the QDagger loss only during the offline phase resulted in much lower performance than using QDagger for both offline and online phase.

A.6 Per-game Scores on ALE

Table A.2: Average scores for the random agent, human agent and the DQN (Adam) trained for 400 million frames (based on 5 runs). For teacher normalized scores reported in the paper, the random agent is assigned a score of 0 while the DQN (Adam) agent is assigned a score of 1. Normalization using teacher allow us to compare the performance of student agents relative to the teacher and avoid high performing outliers (such as Breakout and Space Invaders) in terms of human normalized scores.

Game	Human	DQN @400M	Random
Asterix	8503.3	13682.6	210.0
Beam Rider	16926.5	6608.4	363.9
Bowling	160.7	33.9	23.1
Breakout	30.5	234.2	1.7
Enduro	860.5	1142.0	0.0
Ms Pacman	6951.6	4366.2	307.3
Q∗bert	13455.0	11437.3	163.9
River Raid	17118.0	17061.9	1338.5
Seaquest	42054.7	14228.2	68.4
Space Invaders	1668.7	7613.6	148.0

Table A.3: Per-game scores for the top-performing methods in Figure 2. Mean scores along with minimum and maximum scores, shown in brackets, across 3 runs.

Game	QDagger	Kickstarting	Offline Pretraining
Asterix	12301.1 (9802.2, 14118.4)	7872.0 (6979.4, 8691.9)	4132.2 (3802.6, 4533.1)
BeamRider	6428.3 (6103.2, 6925.3)	5330.9 (4964.8, 5661.2)	1854.5 (1794.6, 1942.4)
Bowling	34.1 (30.0, 38.5)	31.3 (30.0, 33.0)	25.5 (16.4, 30.1)
Breakout	277.6 (254.5, 295.8)	237.6 (191.0, 281.7)	126.3 (118.2, 133.1)
Enduro	1765.6 (1415.1, 2161.3)	1167.3 (1031.9, 1268.2)	1167.9 (1124.7, 1222.9)
MsPacman	5110.6 (4742.7, 5325.6)	4196.8 (4038.8, 4485.4)	3687.2 (3608.7, 3764.3)
Qbert	16198.8 (14957.2, 17521.1)	10028.4 (7337.3, 12860.8)	16421.6 (14018.6, 18395.2)
Riverraid	19496.7 (18585.6, 20068.4)	16160.5 (14474.1, 17768.7)	14137.7 (13858.5, 14681.7)
Seaquest	17056.0 (16560.9, 17743.1)	7849.9 (3674.8, 12504.5)	12099.4 (10374.0, 15204.1)
SpaceInvaders	6891.9 (6055.9, 8374.6)	2459.6 (2111.7, 2726.2)	958.7 (841.1, 1033.2)

Table A.4: Per-game scores for worst-performing methods in Figure 2. Mean scores along with minimum and maximum scores, shown in brackets, across 3 runs.

Game	Rehearsal	DQfD	JSRL
Asterix	2329.6 (2172.0, 2503.9)	2370.8 (2140.9, 2681.0)	1294.2 (1018.2, 1720.3)
BeamRider	2740.4 (2262.7, 3123.5)	2482.9 (1596.4, 3246.7)	4642.0 (4272.0, 5324.4)
Bowling	33.5 (30.0, 36.0)	33.0 (30.0, 39.0)	37.8 (30.0, 46.1)
Breakout	70.2 (56.1, 82.2)	31.2 (21.2, 38.4)	49.7 (35.8, 59.2)
Enduro	985.0 (928.6, 1095.0)	736.6 (671.5, 792.4)	839.4 (660.7, 949.9)
MsPacman	2604.3 (2343.2, 2997.0)	2886.2 (2576.5, 3184.8)	2140.4 (1871.8, 2585.4)
Qbert	10649.0 (7697.8, 13882.9)	13386.7 (11800.7, 15295.5)	5828.7 (2871.6, 7753.4)
Riverraid	8640.5 (7789.1, 9532.1)	10290.7 (9722.8, 11323.5)	7498.6 (6916.7, 7916.8)
Seaquest	2859.2 (2300.9, 3969.2)	5325.5 (4454.8, 6128.3)	1140.8 (760.3, 1590.0)
SpaceInvaders	657.3 (638.7, 689.4)	533.3 (520.1, 557.5)	570.1 (549.9, 600.9)