The Difficulty of Passive Learning in Deep Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning to act from observational data without active environmental interaction is a well-known challenge in Reinforcement Learning (RL). Recent approaches involve constraints on the learned policy or conservative updates, preventing strong deviations from the state-action distribution of the dataset. Although these methods are evaluated using non-linear function approximation, theoretical justifications are mostly limited to the tabular or linear cases. Given the impressive results of deep reinforcement learning, we argue for a need to more clearly understand the challenges in this setting. In the vein of Held & Hein's classic 1963 experiment, we propose the "tandem learning" experimental paradigm which facilitates our in-depth empirical analysis of the difficulties in offline reinforcement learning. We identify function approximation in conjunction with fixed data distributions as the strongest factors, thereby extending but also challenging hypotheses stated in past work. Our results provide relevant insights for offline deep reinforcement learning, while also shedding new light on phenomena observed in the online case of learning control.

1 Introduction

Learning to act in an environment purely from observational data (i.e. with no environment interaction), usually referred to as *offline reinforcement learning*, has great practical as well as theoretical importance (see [25] for a recent survey). In real-world settings like robotics and healthcare, it is motivated by the ambition to learn from existing datasets and the high cost of environment interaction. Its theoretical appeal is that the stationarity of the data distribution allows for more straightforward convergence analysis of learning algorithms. Moreover, decoupling learning from data generation alleviates one of the major difficulties in the empirical analysis of common reinforcement learning agents, allowing the targeted study of learning dynamics in isolation from its effects on behavior.

Recent work has identified *extrapolation error* as a major challenge for offline (deep) reinforcement learning [2, 8, 13, 11, 26], with *bootstrapping* often highlighted as either the cause or an amplifier of the effect: The value of missing or under-represented state-action pairs in the dataset can be over-estimated, either transiently (due to insufficient training or data) or even asymptotically (due to modelling or dataset bias), resulting in a potentially severely under-performing acquired policy. The corrective feedback-loop [24], whereby value over-estimation is self-correcting via *exploitation* during interaction with the environment (while under-estimation is corrected by *exploration*), is critically missing in the offline setting.

To mitigate this, typically one of a few related strategies are proposed: policy or learning update constraints to prevent significant deviations from states and actions well-covered by the given dataset [12, 13, 23, 2, 40, 41], a pessimism bias to battle value over-estimation [8], large and diverse datasets to improve state space coverage [3], or a learned model to fill in gaps with synthesized data [32].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

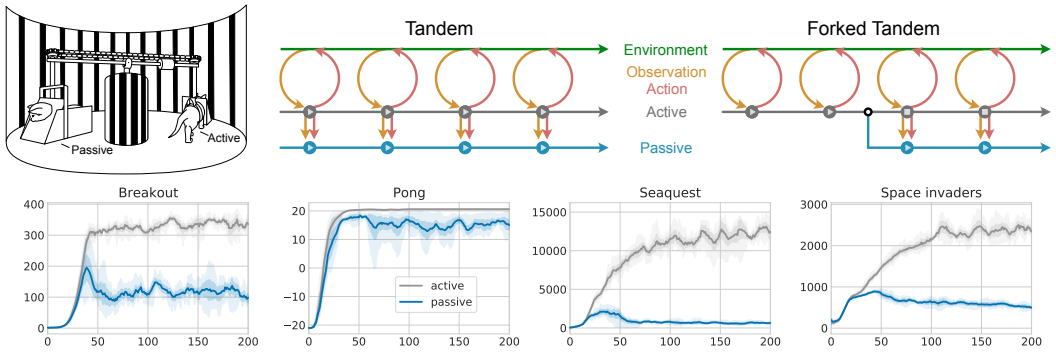

Figure 1: (**top-left**) Held and Hein [15] experiment setup. (**top-right**) Illustrations of the Tandem and Forked Tandem experiment setups. (**bottom**) Tandem (active and passive) performance on 4 selected Atari domains. In all figures, *active* agent performance is shown in **gray**.

While many of these enjoy theoretical justification in the tabular or linear function approximation cases [35], guarantees for the practically relevant non-linear case are mostly lacking.

In this paper we draw inspiration from the experimental paradigm introduced in the classic Held and Hein [15] experiment in psychology. The experiment involved coupling two young animal subjects' movements and visual perceptions in such a way that both receive the same stream of visual inputs, while only one can actively shape that stream by directing the pair's movements (Fig. 1, top-left). By showing that, despite identical visual experiences, only the actively moving subject acquired adequate visual acuity, the experiment established the importance of active locomotion in learning vision. Analogously, here we introduce the 'Tandem RL' setup, which pairs an 'active' and a 'passive' agent in a training loop where only the active agent drives data generation, while both perform identical learning updates from the generated data[1]. By decoupling learning dynamics from its impact on data generation, while preserving the non-stationarity of the online learning setting, the experimental paradigm of tandem learning promises to be a valuable analytic tool for the precise empirical study of RL algorithms.

Holding architectures, losses, and crucially data distribution equal across the active and passive agents, or varying them in a controlled manner, we perform a detailed empirical analysis of the failure modes of 'passive' (i.e. non-interactive, offline) learning, and pinpoint the contributing factors in features of the data distribution, function approximation and learning algorithm. Our study confirms some past intuitions for the failure modes of offline learning, while refining and extending the findings in the deep RL case. In particular, our results indicate an empirically less critical role for bootstrapping than previously hypothesized, while foregrounding over-generalization and extrapolation by function approximation trained on a fixed data distribution as the crucial challenge. Among other things, our experiments draw a sharp boundary between the mostly well-behaved (and analytically well-studied) case of *linear* function approximation, and the *non-linear* case for which theoretical guarantees are lacking. Moreover, we delineate different, more and less effective, ways of enhancing the training data distribution to support successful offline learning, e.g. by analysing the impact of dataset size and diversity, the stochasticity of the data generating policy, or small amounts of self-generated data. Our results provide hints towards a hypothesis relevant in both offline and online RL: robust learning of control with function approximation may require interactivity not merely as a data gathering mechanism, but as a counterbalance to a (sufficiently expressive) approximator's tendency to 'exploit gaps' in an arbitrary fixed data distribution by excessive extrapolation.

2 The Experimental Paradigm of Tandem Reinforcement Learning

The Tandem RL setting, extending a similar analytic setup in [13], consists of two learning agents, one of which (the 'active agent') performs the usual online training loop of interacting with an environment and learning from the generated data, while the other (the 'passive agent') learns solely

[1]The 'tandem' analogy is of course that of two riders, both of whom experience the same route, while only the front rider gets to decide on direction.

from data generated by the active agent, while only interacting with the environment for evaluation. We distinguish two experimental paradigms (see Fig. 1, top-right):

Tandem: Active and passive agents start with independently initialized networks, and train on an identical sequence of training batches in the exact same order.

Forked Tandem: An agent is trained for a fraction of the training budget. It is then 'forked' into active and passive agents, which start out with identical network weights. The active agent is 'frozen', i.e. receives no further training, but continues to generate data from its policy. The passive agent is trained on this generated data.

2.1 Implementation

Our basic experimental agent is 'Tandem DQN', an active/passive pair of Double-DQN agents[2] [38]. Following the usual training protocol [28], the total training budget is 200 iterations, each of which consists of 1M steps taken on the environment by the active agent, interspersed with regular learning updates (on one, or concurrently on both agents, depending on the paradigm), on batches of transitions sampled from the active agent's replay buffer. Both agents are independently evaluated on the environment for 500K steps after each training iteration.

Most of our experiments are performed on the Atari domain [5], using the exact algorithm and hyperparameters from [38]. We use a fixed set of four representative games to demonstrate most of our empirical results, two of which (BREAKOUT, PONG) can be thought of as easy and largely solved by baseline agents, while the others (SEAQUEST, SPACE INVADERS) have non-trivial learning curves and remain challenging. Unless stated otherwise, all results show averages over at least 5 seeds, with confidence intervals indicating variation over seeds. In all comparative plots, **boldface** entries indicate the default Tandem DQN configuration.

2.2 The Tandem Effect

We begin by reproducing the striking observation in [13] that the passive learner is generally unable to adequately learn from the very data stream that is demonstrably sufficient for its architecturally identical active counterpart; we refer to this phenomenon as the 'tandem effect' (Fig. 1, bottom). We ascertain the generality of this finding by replicating it across a broad suite of environments and agent architectures: Double-DQN on 57 Atari environments (Appendix Figs. 9 & 10), adapted agent variants on four Classic Control domains from the OpenAI Gym library [1] and the MinAtar domain [42] (Appendix Figs. 11 & 14), and the distributed R2D2 agent [21] (Appendix Fig. 13). Details on agents and environments are provided in the Appendix.

Empirically, we make the informal observation that while active and passive Q-networks tend to produce similar values for typical state-action pairs under the active policy (where the action is the active Q-value function's argmax for a given state), their values are less correlated for other (non-argmax) actions, and in fact the active and passive greedy policies of a Tandem DQN tend to disagree in a large fraction of states under the behavior distribution (on average $> 75\%$ of states, after 100M steps of training, across 57 Atari games; see Appendix Fig. 12). Moreover, in a substantial fraction ($\approx 12/57$) of Atari games, we observe the passive agent's network to strongly over-estimate a fraction of state-action values, with the over-estimation growing as training progresses.

3 Analysis of the Tandem Effect

In line with existing explanations [25], we propose that the tandem effect is primarily caused by extrapolation error when certain state-action pairs are under-represented in the behavioral data of the active agent. Specifically with ε-greedy policies, even small over-estimation of the values of rarely seen actions can lead to sufficient behavior deviations to cause catastrophic under-performance.

We further extend this hypothesis: in the context of *deep* reinforcement learning (i.e. when using non-linear function approximation), an inadequate data distribution can drive excessive over-generalization

[2] Our choice of Double-DQN as a baseline is motivated by its relatively strong performance and robustness compared to vanilla DQN [28], paired with its simplicity compared to later variants like Rainbow [16], which allows for more easily controlled experiments with fewer moving parts.

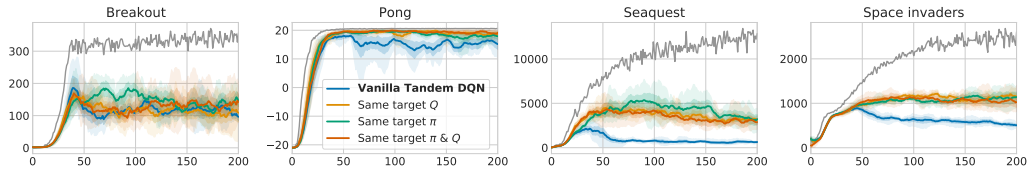

Figure 2: Active vs. passive performance when using the active agent's target policy and/or value function for constructing passive bootstrapping targets.

[6], making such erroneous extrapolation likely. While the tandem effect can show up as learning inefficiency even in the tabular case, it proves especially pernicious in the case of non-linear function approximation, where over-generalization or extrapolation can lead to errors not just on completely unseen, but also rarely seen data, and can persist in the infinite-sample limit.

Coalescing this view and past analyses of challenges in offline RL (e.g. [25, 13, 26]) into the following three explanatory hypotheses about the tandem effect provides a natural structure to our analysis:

Hypothesis 1 (Bootstrapping) *The tandem effect is caused by the passive agent bootstrapping from poorly estimated (in particular, over-estimated) values: this allows an initially small divergence from the active network to get amplified.*

Hypothesis 2 (Data Distribution) *The tandem effect is caused by the insufficient coverage of sub-optimal actions (under the data generating policy). Other confounding factors, like the stationarity of the data distribution, play only a small role.*

Hypothesis 3 (Function Approximation) *The tandem effect is caused by the use of a non-linear function approximator as a Q-value function, which erroneously extrapolates the values of state-action pairs infrequently seen under the active behavior distribution, despite their relevance under the passively learned policy. This results in under-performance when evaluated for control. Moreover, this instability is inherent, in the sense of being independent of initialisation.*

3.1 The Role of Bootstrapping

One distinguishing feature of reinforcement learning as opposed to supervised learning is its frequent use of learned quantities as preliminary optimization targets, most prominently in what is referred to as *bootstrapping* in the widely used TD algorithms [33], where preliminary estimates of the value function are used as update targets. In the Double-DQN algorithm these updates take the form $Q(s,a) \leftarrow r + \gamma \bar{Q}(s', \arg\max_{a'} Q(s',a'))$, where Q denotes the parametric Q-value function, and $\bar{Q}$ is the target network Q-value function, i.e. a time-delayed copy of Q.

Four value functions are involved in the active and passive updates of Tandem DQN: $Q_A, \bar{Q}_A, Q_P$ and $\bar{Q}_P$, where the A/P subscripts refer to the Q-value functions of the active and passive agents, respectively. The use of its own target network by the passive agent makes Hypothesis 1 appear highly plausible. To test this, we replace the target values and/or policies in the update equation for the *passive* agent, with the values provided by the *active* agent's value functions:

$$Q_P(s,a) \leftarrow \begin{cases} r + \gamma \bar{Q}_P(s', \arg\max_{a'} Q_P(s',a')) & \text{Vanilla Tandem DQN} \\ r + \gamma \bar{Q}_A(s', \arg\max_{a'} Q_P(s',a')) & \text{Same Target } Q \\ r + \gamma \bar{Q}_P(s', \arg\max_{a'} Q_A(s',a')) & \text{Same Target } \pi \\ r + \gamma \bar{Q}_A(s', \arg\max_{a'} Q_A(s',a')) & \text{Same Target } \pi \& Q \end{cases}$$

As shown in Fig. 2, the use of the active value functions as targets reduces the active-passive gap by only a small amount. Note that when both active target values and policy are used, both networks are receiving *an identical sequence of targets* for their update computations, a sequence that suffices for the active agent to learn a successful policy. Strikingly, despite this the tandem effect appears largely preserved: in all but the easiest games (e.g. PONG) the passive agent fails to learn effectively.

This finding **weakens Hypothesis 1**: bootstrapping does not appear to be a major cause of the tandem effect, it is (at most) amplifying it. It also points to data distribution and function approximator

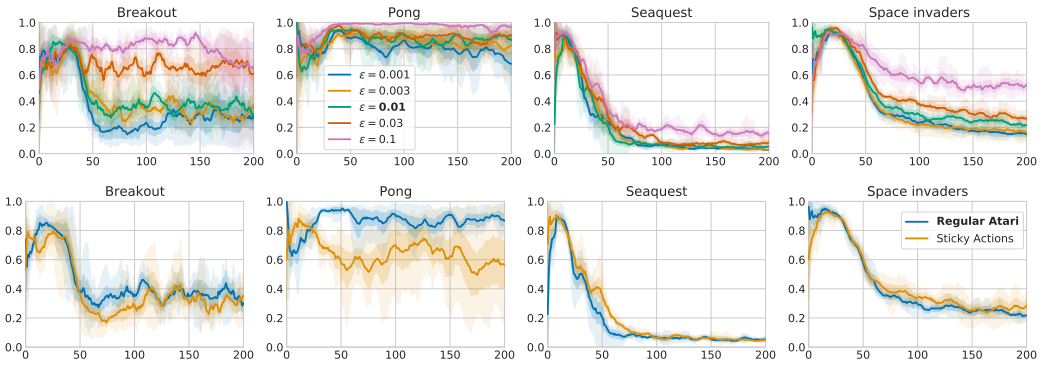

Figure 3: Passive as fraction of active performance for varying active ε-greedy behavior policies (**top**); regular Atari vs sticky-actions Atari (**bottom**). We report *relative* passive performance, as active performance varies across configurations. See Appendix Figs. 15 & 16 for absolute performance.

(unchanged in these experiments) as the likelier causal factors. Additional evidence for this is provided below, where the tandem effect occurs in a purely supervised setting without bootstrapping.

3.2 The Role of the Data Distribution

The critical role of the data distribution for offline learning is well established [13, 20, 26]. Here we extend past analysis and provide new insight into this connection, by investigating how properties of the data distribution (e.g. stochasticity, stationarity, or the size and diversity of the dataset) affect its suitability for passive learning.

The exploration parameter ε A simple way to affect the data distribution's state-action coverage (albeit in a blunt and uniform way) is by varying the exploration parameter ε of the active agent's ε-greedy behavior policy (for training, not for evaluation). Note that a higher ε parameter affects the *active* agent's own training performance, as its ability to navigate environments requiring precise control is reduced. We therefore report the *relative* passive performance (i.e. as a fraction of the active agent's performance, which itself also varies across parameters). As can be seen in Fig. 3 (top), relative passive performance is indeed substantially improved when the active behavior policy's stochasticity (and as a consequence its coverage of non-argmax actions along trajectories) is increased, and conversely it reduces with a greedier behavior policy, **as predicted by Hypothesis 2**.

Sticky actions An alternative potential source of stochasticity is the environment itself, e.g. the use of 'sticky actions' in Atari [27]: with fixed probability, an agent action is ignored (and the previous action repeated instead). This type of environment-side stochasticity should not be expected to cause new actions to appear in the behavior data, and indeed Fig. 3 (bottom) shows no substantial impact on the tandem effect, providing further **support for Hypothesis 2**.

Replay size Our results contrast with the strong offline RL results in [3]. We hypothesize that the difference is due to the vastly different data set size (full training of 200M transitions vs. replay buffer of 1M). Interpolating between the tandem and the offline RL setting from [3], we increase the replay buffer size, thereby giving the passive agent access to somewhat larger data diversity and state-action coverage. **In line with Hypothesis 2**, a larger replay buffer somewhat mitigates the passive agent's under-performance (Fig. 4), though it appears to mostly slow down rather than prevent the passive agent from eventually under-performing its active counterpart substantially. As we suspect that a sufficient replay buffer size may depend on the effective state-space size of an environment, we also perform analogous experiments on the (much smaller) classic control domains; results (Appendix Fig. 20) remain qualitatively the same.

Note that for a fixed dataset size, sample diversity can take different forms. Many samples from a single policy may provide better coverage of states on, or near, policy-typical trajectories. Meanwhile, a larger collection of policies, with fewer samples per policy, provides better coverage of many trajectories at the expense of lesser coverage of small deviations from each. To disentangle the impact

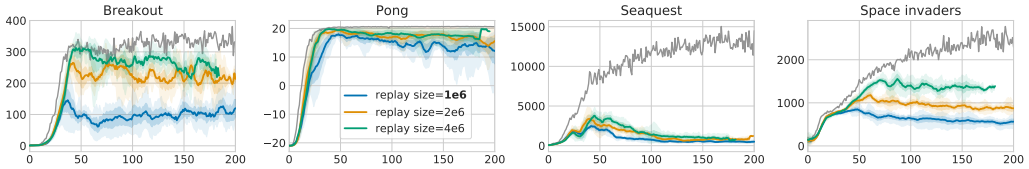

Figure 4: Active vs. passive performance for different replay sizes (for passive agent).

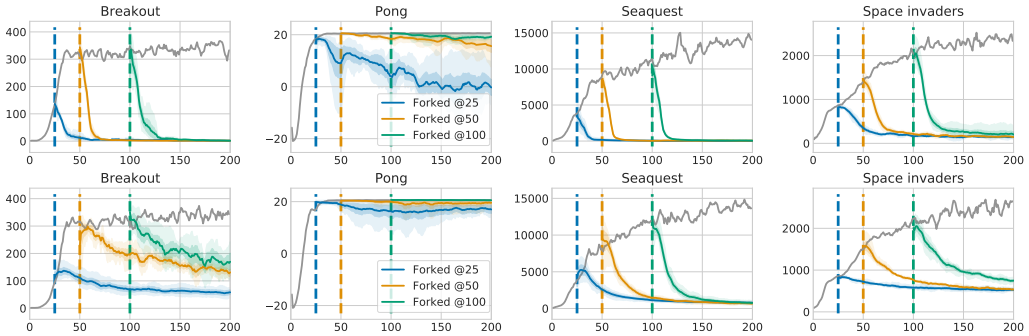

Figure 5: Performance of a Forked Tandem DQN, training passively after freezing its data generating policy (**top**) or its replay buffer (**bottom**). Vertical lines indicate forking time points.

of these modalities, while also shedding light on the role of stationarity of the distribution, we next switch to the 'Forked Tandem' variation of the experimental paradigm.

Fixed policy Upon forking, the frozen active policy is executed to produce training data for the passive agent, which begins its training initialized with the active network's weights. Note that this constitutes a stronger variant of the tandem experiment. At the time of forking, the agents do not merely share analogous architectures and equal 'data history', but also identical network weights (whereas in the simple tandem setting, the agents were distinguished by independently initialized networks). Moreover, the data used for passive training can be thought of as a 'best-case scenario': generated by a single fixed policy, identical to the passive policy at the beginning of passive training. Strikingly, the tandem effect is not only preserved but even exacerbated in this setting (Fig. 5, top): after forking, passive performance decays rapidly in all but the easiest games, despite starting from a near-optimally performing initialization. This re-frames the tandem effect as not merely the difficulty of passively *learning to act*, but even to passively *maintain performance*. Instability appears to be inherent in the very learning process itself, **providing strong support to Hypotheses 2 and 3**.

In Appendix Fig. 21 we additionally show that similarly to the regular tandem setting, stochasticity of the active policy *after forking* strongly influences the passive agent's ability to maintain performance.

Fixed replay A variation on the above experiments is to freeze the *replay buffer* while continuing to train the passive policy from this fixed dataset. Instead of a stream of samples from a single policy, this fixed data distribution now contains a fixed number of samples from a training process of the length of the replay buffer, i.e. from a number of different policies. The collapse of passive performance here (Fig. 5, bottom) is less rapid, yet qualitatively similar. In Appendix Fig. 22 we present yet another variant of this experiment with similar results, showing that the effect is robust to minor variations in the exact way of fixing the data distribution of a learning agent.

These experiments provide **strong evidence for Hypothesis 2**: a larger replay buffer, containing samples from more diverse policies, can be expected to provide an improved coverage of (currently) non-greedy actions, reducing the tandem effect. While the forked tandem begins passive learning with the seemingly advantageous high-performing initialization, the most relevant variable of state-action coverage is critically limited here. In the frozen-policy case, a large number of samples from the very same ε-greedy policy can be expected to provide very little such coverage of non-greedy actions, while in the frozen-replay case, a smaller number of samples from multiple policies can be expected to only do somewhat better in this regard. In both cases the tandem effect is highly pronounced.

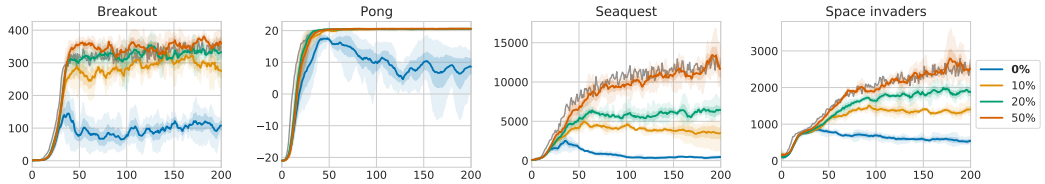

Figure 6: Passive performance for different amounts of self-generated data in the passive agent's replay batches.

On-policy evaluation The strength of the last two experiments lies in the observation that, since active and passive networks have identical parameter values at the beginning of passive training, their divergence cannot be attributed to small initial differences getting amplified by training on an inadequate data distribution. With so many factors held fixed, the catastrophic collapse of passive performance when trained on the very data distribution produced by *its own initial policy* begs the question whether off-policy Q-learning itself is to blame for this failure mode, e.g. via the statistical overestimation bias introduced by the $\max$ operator [14]. In Appendix A.3 we provide a negative answer, by performing on-policy evaluation with SARSA [31], and even purely supervised regression on the Monte-Carlo returns in the forked tandem setup. While evaluation succeeds, in the sense of minimizing evaluation error on the given behavior distribution, atypical action values under the behavior policy suffer substantial estimation error, resulting in occasional over-estimation. The resulting ε-greedy control policy under-performs the initial policy at forking time as catastrophically as in the other forking tandem experiments (Appendix Figs. 23 & 24). **Supporting Hypotheses 2 and 3 while further weakening Hypothesis 1**, these observations point to an inherent instability of offline learning, different from that of Baird's famous example [4] or the 'Deadly Triad' [34, 39]; an instability that results purely from erroneous extrapolation by the function approximator, when the utilized data distribution does not provide adequate coverage of relevant state-action pairs.

Self-generated data Our final empirical question in this section is 'How much data *generated by the passive agent* is needed to correct for the tandem effect?'. While a full investigation of this question exceeds the scope of this paper and is left for future work, the tandem setup lends itself to a simple experiment: *both* agents interact with the environment and fill individual replay buffers, one of them (for simplicity still referred to as 'passive') however learns from data stochastically mixed from both replay buffers. Fig. 6 shows that just a moderate amount (10%-20%) of 'own' data yields a substantial reduction of the tandem effect, while a 50/50 mixture completely eliminates it.

3.3 The Role of Function Approximation

We structure our investigation of the role of function approximation in the tandem effect into two categories: the *optimization* process and the *function class* used.

Optimization Agarwal et al. [3] and Obando-Ceron and Castro [29] demonstrated that the ADAM optimization algorithm [22] often outperforms the RMSProp optimizer [36] used in our experiments. In Appendix Fig. 18 we show that despite better performance of both active and passive agents with ADAM, the tandem effect itself is independent of the choice of optimizer.

Another plausible hypothesis is that the passive network suffers from under-fitting and requires more updates on the same data to attain comparable performance to the active learner. Varying the number of passive agent updates per active agent update step, we find that more updates *worsen* the performance of the passive agent (Appendix Fig.19). This rejects insufficient training as a possible cause, and further **supports Hypothesis 2**.

Function class Given that the active and passive agents share an identical network architecture, the passive agent's under-performance cannot be explained by an insufficiently expressive function approximator. Performing the tandem experiment with pure regression of the passive network's outputs towards the active network's (a variant of *network distillation* [18]), instead of TD based training, we observe that the performance gap is indeed vastly reduced and in some games closed entirely (see Appendix Fig. 17); however, strikingly, it remains in some.

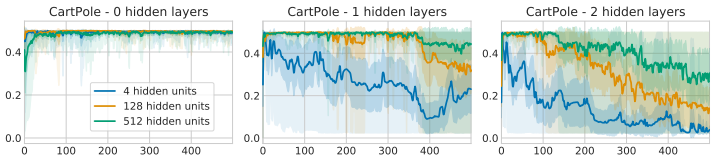

Figure 7: Passive performance as a fraction of active performance in CartPole: varying number of hidden layers and units.

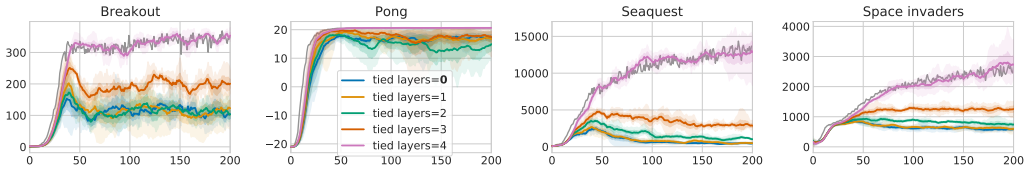

Figure 8: Active vs. passive performance, with the first k of 5 layers of active/passive networks shared.

Next, we vary the function class of both networks by varying the depth and width of the utilized Q-networks on a set of Classic Control tasks. As can be seen in Fig. 7 (and Appendix Fig. 26), the magnitude of the active-passive performance gap appears negatively correlated with network width, which **supports Hypothesis 3:** an increase in network capacity results in less pressure towards over-generalizing to infrequently seen action values and an ultimately smaller tandem effect. On the other hand, the gap seems to correlate *positively* with network depth. We speculate that this may relate to the finding of Huh et al. [19], that deeper networks tend to be biased towards simpler (e.g. lower rank) solutions which may suffer from increased over-generalization.

Finally, we investigate varying the function class of *only the passive network* by sharing the weights of the first k (out of 5) layers of active and passive networks, while constraining the passive network to only update the remaining top $5 - k$ layers, and using the 'representation' at layer k acquired through active learning only. This reduces the 'degrees of freedom' of the passive agent, which we hypothesize reduces its potential for divergence. Indeed, Fig. 8 illustrates that passive performance correlates strongly with the number of *tied* layers, with the variant for which only the linear output layer is trained passively performing on par with the active agent. A similar result is obtained in the forked tandem setting, see Appendix Fig. 25. This finding provides a strong indirect **hint towards Hypothesis 3**: with only part of the network's layers being trained passively, much of its 'generalization capacity' is shared between active and passive agents. States that are not aggregated by the shared bottom layers (only affected by active training) have to be 'erroneously' aggregated by the remaining top layers of the network for over-generalization to occur. A more thorough investigation of this, exceeding the scope of this paper, may involve attempting to measure (over-)generalization more directly, e.g. via *gradient interference* [6].

4 Applications of the Tandem Setting

In addition to being valuable for studying the challenges in offline RL, we propose that the Tandem RL setting provides analytic capabilities that make it a useful tool in the empirical analysis of general (online) reinforcement learning algorithms. At its core, the tandem setting aims to decouple learning dynamics from its impact on behavior and the data distribution, which are inseparably intertwined in the online setting. While classic offline RL achieves a similar effect, as an analytic tool it has the potential downside of typically using a stationary distribution. Tandem RL, on the other hand, presents the passive agent with a data distribution which realistically represents the type of non-stationarity encountered in an online learning process, while still holding that distribution independent from the learning dynamics being studied. This allows Tandem RL to be used to study, e.g., the impact of variations in the learning algorithm on the quality of a learned representation, without having to control for the indirect confounding effect of a different behavior causing a different data distribution.

While extensive examples of this exceed the scope of this paper, in Appendix A.5 we provide a single such experiment, testing QR-DQN [10] as a passive learning algorithm (with Double-DQN as the

active agent). This is motivated by the observation of Agarwal et al. [3], that QR-DQN outperforms DQN in the offline setting. As seen in Fig. 27, QR-DQN indeed is a nontrivially *different* passive learning algorithm, significantly better in some, while curiously worse in other domains.

5 Discussion and Conclusion

At a high level, our work can be viewed as investigating the issue of (in)compatibility between the data distribution used to train a function approximator and the data distribution relevant in its evaluation. While in supervised learning, *generalization* can be viewed as the problem of transfer from a training to a (given) test distribution, the fundamental challenge for control in reinforcement learning is that the test distribution is *created* by the very outcome of learning itself, the learned policy. The various difficulties of learning to act from offline data alone throw into focus the role of interactivity in the learning process: only by continuously interacting with the environment does an agent gradually 'unroll' the very data on which its performance will be evaluated.

This need not be an obstacle in the case of *exact* (i.e. tabular) functions: with sufficient data, extrapolation error can be avoided entirely. In the case of function approximation however, as small errors compound rapidly into a difference in the underlying state distribution, significant divergence and, as this and past work demonstrates, ultimately catastrophic under-performance can occur. Function approximation plays a two-fold role here: (1) being an approximation, it allows deviations in the *outputs*; (2) as the learned quantity, it is (especially in the non-linear case) highly sensitive to variations in the *input distribution*. When evaluated for control after offline training, these two roles combine in a way that is 'unexplored' by the training process: minor *output* errors cause a drift in behavior, and thereby a drift in the test distribution.

While related, this challenge is subtly different from the well-known divergence issues of off-policy learning with function approximation, demonstrated by Baird's famous counterexample [4] (see also [37]) and conceptualized as the Deadly Triad [34, 39]. While these depend on bootstrapping as a mechanism to cause a feedback-loop resulting in value divergence, our results show that the offline learning challenge persists even without bootstrapping, as small differences in behavior cause a drift in the 'test distribution' itself. Instead of a *training-time* output drift caused by bootstrapping, the central role is taken by a *test-time* drift of the state distribution caused by the interplay of function approximation and a fixed data distribution (as opposed to dynamically self-generated data).

Our empirical work highlights the importance of interactivity and 'learning from your own mistakes' in learning control. Starting out as an investigation of the challenges in offline reinforcement learning, it also provides a particular view point on the classical online reinforcement learning case. Heuristic explanations for highly successful deep RL algorithms like DQN, based on intuitions from (e.g.) approximate policy iteration, need to be viewed with caution in light of the apparent hardness of a policy improvement step based on approximate policy evaluation with a function approximator.

Finally, the forked tandem experiments show that even high-performing initializations are not robust to a collapse of control performance, when trained under their own (*but fixed!*) behavior distribution. Not just 'learning to act', but even 'maintaining performance' appears hard in this setting. This provides an intuition that we distill into the following working conjecture: *The dynamics of deep reinforcement learning for control are unstable on (almost) any **fixed** data distribution.*

Expanding on the classical on- vs. off-policy dichotomy, we propose that indefinitely training on *any fixed* data distribution, without explicit regularization or additional inductive bias, gives rise to an 'exploitation of gaps in the data' by a function approximator, akin to the over-fitting occurring when over-training on a fixed dataset in supervised learning. Interaction, i.e. generating at least moderate amounts of one's own experience, appears to be a powerful, and for the most part *necessary*, regularizer and stabilizer for learning to act, by creating a dynamic equilibrium between optimization of a function approximator and its own data-generation process.

Broader impact statement

This work lies in the realm of "foundational RL" in that it contributes to the fundamental understanding and development of reinforcement learning algorithms and theory. As such, our work is far removed from ethical issues and potential societal consequences. We do not propose new algorithms, but instead study, in an attempt at scientific understanding, the empirical phenomena present in various forms of offline reinforcement learning.

References

[1] OpenAI gym. https://github.com/openai/gym.

[2] Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep q-learning. *CoRR*, abs/1903.08894, 2019.

[3] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

[4] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 30–37. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50013-X.

[5] 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.

[6] Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In *International Conference on Machine Learning*, pages 767–777. PMLR, 2020.

[7] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[8] Jacob Buckman, Carles Gelada, and Marc G Bellemare. The importance of pessimism in fixed-dataset policy optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=E3Ys6a1NTGT.

[9] David Budden, Matteo Hessel, John Quan, Steven Kapturowski, Kate Baumli, Surya Bhupatiraju, Aurelia Guy, and Michael King. RLax: Reinforcement Learning in JAX, 2020. URL http://github.com/deepmind/rlax.

[10] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[11] Rasool Fakoor, Jonas Mueller, Pratik Chaudhari, and Alexander J Smola. Continuous doubly constrained batch reinforcement learning. *arXiv preprint arXiv:2102.09225*, 2021.

[12] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *CoRR*, abs/1910.01708, 2019. URL http://arxiv.org/abs/1910.01708.

[13] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 09–15 Jun 2019.

[14] Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

[15] R. Held and A. Hein. Movement-produced stimulation in the development of visually guided behavior. *Journal of Comparative and Physiological Psychology*, 56(5):872–876, 1963.

[16] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[17] Matteo Hessel, David Budden, Fabio Viola, Mihaela Rosca, Eren Sezener, and Tom Hennigan. Optax: Composable gradient transformation and optimisation, in JAX!, 2020. URL http://github.com/deepmind/optax.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

[19] Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *CoRR*, abs/2103.10427, 2021. URL https://arxiv.org/abs/2103.10427.

[20] Alexis Jacq, Matthieu Geist, Ana Paiva, and Olivier Pietquin. Learning from a learner. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2990–2999. PMLR, 09–15 Jun 2019.

[21] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

[22] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *Proceedings of the International Conference on Learning Representations*, 2015.

[23] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[24] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18560–18572. Curran Associates, Inc., 2020.

[25] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL https://arxiv.org/abs/2005.01643.

[26] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

[27] 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.

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

[29] Johan S Obando-Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2021.

[30] John Quan and Georg Ostrovski. DQN Zoo: Reference implementations of DQN-based agents, 2020. URL http://github.com/deepmind/dqn_zoo.

[31] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[32] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *arXiv preprint arXiv:2104.06294*, 2021.

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

[34] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[35] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, pages 2380–2388. PMLR, 2015.

[36] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5: Rmsprop. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[37] John N. Tsitsiklis and Benjamin Van Roy. Analysis of temporal-difference learning with function approximation. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, page 1075–1081, Cambridge, MA, USA, 1996. MIT Press.

[38] 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.

[39] Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

[40] Ziyu Wang, Alexander Novikov, Konrad Żołna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

[41] Yue Wu, Shuangfei Zhou, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2021.

[42] Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

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]
   (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] Code provided as supplemental.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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] Provided in the supplemental.
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]

(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]