The Phenomenon of Policy Churn

Anonymous Author(s)
Affiliation
Address
email

Abstract

We identify and study the phenomenon of *policy churn*, that is, the rapid change of the greedy policy in value-based reinforcement learning. Policy churn operates at a *surprisingly rapid* pace, changing the greedy action in a large fraction of states within a handful of learning updates (in a typical deep RL set-up such as DQN on Atari). We characterise the phenomenon empirically, verifying that it is not limited to specific algorithm or environment properties. A number of ablations help whittle down the plausible explanations on why churn occurs to just a handful, all related to deep learning. Finally, we hypothesise that policy churn is a beneficial but overlooked form of *implicit exploration* that casts ϵ-greedy exploration in a fresh light, namely that ϵ-noise plays a much smaller role than expected.

1 The Phenomenon

Reinforcement learning (RL) involves agents that incrementally update their policy. This process is driven by the objective of maximising reward, and based on experience that the agent generates via exploration. The sequence of policies $\pi_0, \ldots, \pi_k, \ldots, \pi_T$ usually starts from a randomly initialised policy π_0 and aims to end at a near-optimal policy $\pi_T \approx \pi^*$. Ideally, steps in that sequence ($\pi_k \to \pi_{k+1}$) are policy improvements that increase expected reward.

This paper studies the amount of *policy change* that goes along with such a policy update process (for a definition, see Section 1.1). In particular, it makes the core observation that policy change *in practice* (as illustrated in some typical deep RL settings) is orders of magnitude larger than could have been expected, and stands in contrast to various reference algorithms (Sections 1.2 and 3.3).

Key observation 1: The greedy policy changes much more rapidly than you probably think.[a]

[a] As a coarse magnitude for the impatient reader: in a typical run of DQN on Atari, the greedy policy changes in $\approx 10\%$ of all states after *a single gradient update* (Figure 1 and Section 1.2).

We dub this phenomenon "*policy churn*", to highlight that most of this policy change may be unnecessary. We study the phenomenon in depth, determining the range of deep RL scenarios it appears in, fleshing out its properties, and in the process narrowing the space of potential causes and mechanisms involved using a set of ablations (Section 3).

Our second key message relates the phenomenon of churn to exploration, specifically in the context of ϵ-greedy exploration (Section 2), with some more speculative ramifications in Section 4.

Key observation 2: Policy churn is a significant driver of exploration.[a]

[a] This holds both in the sense that reducing churn reduces performance, and in the sense that explicitly adding noise becomes unnecessary in the presence of churn (i.e., $\epsilon = 0$ is viable).

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

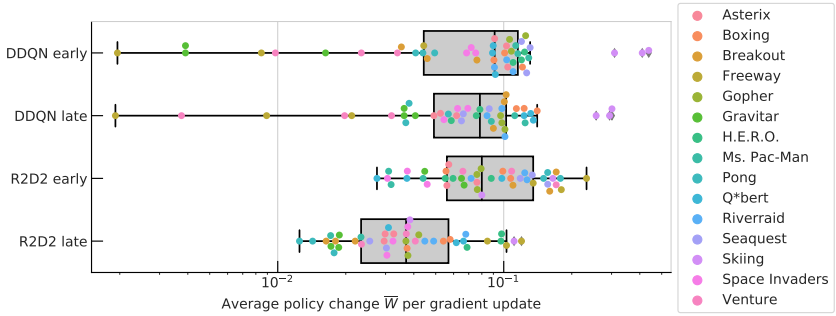

Figure 1: Average amount of policy change $\overline{W}$ per update in two deep RL agents (DoubleDQN and R2D2). Points are averages per seed on one of 15 (colour-coded) Atari games. "Early" denotes the first 25% of training, "late" denotes the final 25%; observe that churn magnitudes drop late in training, but only for R2D2. See Figures 5 and 6 in the appendix for more fine-grained results.

1.1 Defining policy change

A policy is a function from states $s \in \mathcal{S}$ to a distribution over actions $a \in \mathcal{A}$, where for the purposes of this paper $\mathcal{A}$ is discrete. We quantify the local, per-state *policy change* between policies π and π' using the moved probability mass (i.e., the total variation distance):

$$W(\pi, \pi'|s) := \frac{1}{2} \sum_a |\pi(a|s) - \pi'(a|s)| \,, \tag{1}$$

which satisfies $0 \leq W(\pi, \pi'|s) \leq 1$. When π and π' are *greedy* policies derived from a state-action-value function—that is, $\pi \in \arg\max_a q(s, a)$—then $W(\pi, \pi'|s) = 1$ if the $\arg\max$ action in state s changes upon replacing q by the function q' underlying π', and $W(\pi, \pi'|s) = 0$ otherwise. Similar reasoning applies more generally when both π and π' are deterministic policies. We can aggregate policy change across all states, weighted by a state-distribution μ:

$$\overline{W}(\pi, \pi') := \mathbb{E}_{s\sim\mu}[W(\pi, \pi'|s)],$$

where a reasonable choice for μ is the empirical state distribution encountered during training (which is non-stationary, but it could also be the stationary distribution of a fixed policy, or the uniform distribution across all states, as discussed for some of the toy scenarios below). For greedy policies, $\overline{W}(\pi, \pi')$ is simply the fraction of states where an $\arg\max$ switch occurred.

For settings where policy performance stabilises at some point $t = P$ of training (e.g., hitting a performance plateau, or converging to optimal behaviour), two additional metrics may be of interest, namely the *cumulative* policy change $\overline{W}_{0:P}$ until that point[1], and the *average* policy change after that point $\overline{W}^+$ (which could be zero, e.g., if the process converges):

$$\overline{W}_{0:P} := \sum_{i=0}^{P-1} \overline{W}(\pi_i, \pi_{i+1}) \qquad\qquad \overline{W}^+ := \lim_{T\to\infty} \frac{1}{T-P} \overline{W}_{P:T} \,.$$

1.2 Quantifying the phenomenon

Given an initial and a final policy, the process with the minimum amount of policy change is an oracle that jumps from π_0 to π_P in a single step ($P = 1$). By construction, this can incur a policy change of at most one unit, $\overline{W}_{0:P} \leq 1$, which is a form of lower bound. In value-based deep RL agents, a natural definition for the sequence of policies is to use the induced greedy policies $\pi_k \in \arg\max_a q_{\theta_k}(s, a)$ where θ_k are the parameters of the Q-function at iteration k. In agents that use a target network

[1]Note that the granularity of updates (e.g., batch size) determines how many intermediate policies are considered, which in turn affects the measured magnitude of policy change in learning processes that have churn.

(inducing π) that is an older copy of the online network (inducing π'), it is easy to measure $\overline{W}(\pi, \pi')$ by comparing their $\arg\max$ actions at the points in training where the target network lags behind by just one update. Figure 1 shows typical values for $\overline{W}$ on a few Atari games, estimated by comparing online and target network policies on batches of experience sampled from the agent's replay buffer. It is worth emphasising that with such rates of average policy change, of $\overline{W} \approx 10\%$ per update, the magnitude of whole-lifetime change becomes enormous: across training, an agent like DQN, which performs 10^7 updates, changes its greedy action *a million times in each state* (on average).

A second striking result is the amount of policy change in *late* training, when the performance of the policy no longer changes (in the case of PONG it is arguably optimal): there is still a change of $\overline{W}^+ \approx 5\%$ per update (Figures 1 and 7). This highlights that a lot (maybe most) of policy churn is not directed at a policy improvement; we revisit this in Section 4.2.

Is this unexpected? We conducted an informal survey of over 50 deep RL practitioners, including some of the inventors of DQN [27], for their estimate on how rapidly the policy changes in a typical Q-learning based setup. The question was: `For the greedy policy to change in 10% of all states, how many learning updates does it take?` (or equivalent). The median response was $1\,000$ updates, with answers varying between 1 and $1\,000\,000$ updates. This deviates by three orders of magnitude from the empirical value of 1 update (or $\overline{W} \approx 0.1$, see Figure 1).

Policy change in other settings. To appreciate the large empirical magnitudes of (cumulative) policy change we observe in deep RL, we can contrast them with a few alternative settings. For example, classic dynamic programming techniques such as value iteration or policy iteration [44], when applied to toy RL domains such as FourRooms, Catch, or DeepSea [33], accumulate $\overline{W}_{0:P} \approx 1$, not much more than the single-step oracle (see Appendix B.4). The two main differences from deep RL are their tabular nature (no function approximation, FA) and non-incremental updates. It is possible to construct tabular settings with much larger policy change, with either incremental updates (Appendix A.3) or bootstrapping (Appendix A.4). A minimalist example with non-linear FA and incremental updates is supervised learning on the MNIST dataset (the "policy" in this case are the predicted label probabilities). Training a digit classifier to convergence accumulates $\overline{W}_{0:P} \approx 10$, that is, the average input goes through 10 label switches (see Appendix B.5). None of these examples are fully satisfying, as they are not apples-to-apples comparisons; so in Section 3.3 we construct a spectrum of algorithmic variants that spans from tabular policy iteration (without churn), via tabular Q-learning to an approximation of DQN (with realistic magnitudes of churn).

2 The Exploration Effect

The next question is what happens when *acting* with the policies produced by a learning process with high churn: it is likely that the rapid change of greedy policies impacts the exploratory behaviour of agents. While each individual greedy policy would lead to a very narrow set of experience in a (nearly) deterministic environment like Atari, the fact that the greedy policy changes so rapidly (in 10% of states per update, with one update every 16 environment frames in DoubleDQN) makes the data distribution broad enough that in many circumstances, no other form of exploration (such as stochasticity introduced via an ϵ-greedy policy) is needed for good performance. To see this effect in practice across a range of Atari games, compare **green** (baseline) and **gold** ($\epsilon = 0$) curves in Figure 2.[2]

Conversely, removing (some of the) churn in the behaviour during training, which can be done by acting with the target network (updated only every $120\,000$ frames in the DoubleDQN agent) instead of the online network, sometimes reduces performance even in the presence of ϵ-greedy exploration

[2] In line with prior work, all our DoubleDQN experiments preserve the decaying ϵ-schedule in the first 2% of training, it is only zero after that initial phase; R2D2 experiments have no such schedule.

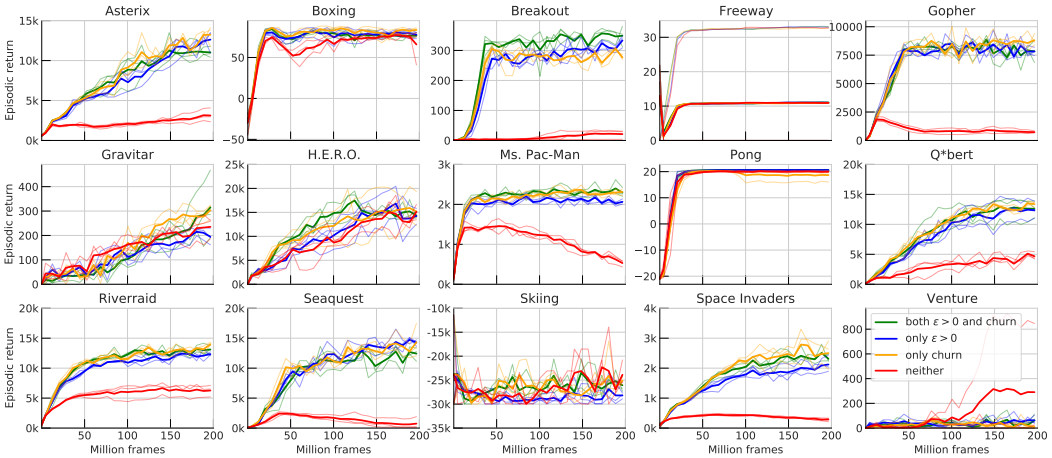

Figure 2: Impact of churn on exploration. Plots show performance of DoubleDQN on Atari, with four variants of exploration: **green** is the unmodified baseline ($\epsilon = 0.01$), **gold** changes $\epsilon = 0$, **blue** has reduced churn by acting with the target network, and **red** shows the effect of both of these changes together (no more exploration, and a corresponding performance collapse in most games). Thin lines depict individual seeds (3 per setting), thick lines are averages, smoothed over 10M frames.

(**blue** in Figure 2).[3] Additionally, we show that performance often *collapses* completely when both forms of exploration are removed (no churn, $\epsilon = 0$, in **red**). Figure 2 compares all four variants of exploration, indicating that the two sources of exploration have different contributions in different games.

Sufficient exploration with $\epsilon = 0$. The perhaps unintuitive observation of successful exploration with purely greedy policies has been made before, albeit implicitly. In particular, in the presence of certain alternative exploration methods such as noisy nets [15], no significant additional advantage is obtained from using $\epsilon > 0$ [20]. Other works containing experimental variants with $\epsilon = 0$ demonstrated successful training in this setting [35, 39] without highlighting the result.

Consistent behaviour and Thompson sampling. In considering the potential exploration benefits of a rapidly changing policy, it is worth qualitatively contrasting the resulting behaviour to that of an ϵ-greedy policy with $\epsilon > 0$. The latter generates high-frequency dithering, with uncorrelated random action decisions at consecutive states and the effect of most exploratory actions likely undone by the following greedy action [11]. By contrast, policy churn induced exploration can be expected to generate temporally correlated, consistent exploration (necessary, though perhaps not sufficient, to perform "deep" exploration, as in ensemble and Thompson sampling methods [45, 32]). On the other hand, while ϵ-greedy exploration is explicitly unbiased in action space, policy churn likely prefers exploration across near-optimal actions (with respect to the current value function). This may be beneficial in some settings, for example when some actions are deadly, and high ϵ prevents long episodes. It may also be detrimental in others, where $\epsilon > 0$ helps the agent to avoid getting stuck.

3 Potential Causes

With the presence and impact of the churn phenomenon established, this section aims to provide additional depth. First, we look at the generality of the effect in Section 3.1. Second, we conduct investigations into the sensitivity of the phenomenon, with Section 3.2 showing ablations to the large-scale deep RL agents (with more material in Appendix A), and Section 3.3 taking the complementary

[3] Acting with the target network also introduces *latency* on how fast newly learned knowledge can be exploited. To see how specifically this latency should have a negligible effect on performance, imagine shifting the x-axis of the blue curve by 120 000 frames to the left, which would be an oracle "target-network-of-the-future" variant.

Agent	DoubleDQN	R2D2
Input	84×84 grayscale	210×160 RGB
Action set	minimal per game	full ($\lvert\mathcal{A}\rvert = 18$)
Reward	clipped	unclipped
Neural net	feed-forward, 1.7M parameters	recurrent, 5.5M parameters
Q-value head	regular	dueling
Update	1-step double Q-learning	5-step double Q-learning
Optimiser	RMSProp w/o momentum	Adam with momentum $= 0.9$
Batch size	32	$32 \times 80 = 2560$
Replay, replay ratio	uniform, 8	prioritised (exponent 0.9), 1
Parallel actors	1	192
Mean $\overline{W}$ per update	$\approx 9\%$	$\approx 6\%$

Table 1: Our two considered agent setups differ in a number of properties. Given both settings result in similar churn, none of them seem critical. See Appendix B.1 and B.2 for details.

approach of interpolating between dynamic programming and a DQN approximation on a toy domain. Finally, Section 3.4 synthesises the findings and postulates some compatible underlying mechanisms.

3.1 Breadth of prevalence and non-causes

To judge the importance of the phenomenon, we need to establish whether it is specific to a narrow range of settings, or prevalent in a variety of domains, algorithmic variants, and hyper-parameters. It turns out that this is easy to do, because the effect is very much not a subtle one. In fact, policy churn is present in two very different deep RL agents, namely DoubleDQN [47] and (a variant of) R2D2 [21], both widely used for training on Atari (see Appendix B.1 and B.2 for agent and environment specifications). Despite the large differences between the algorithms summarised in Table 1, the magnitude of policy change is surprisingly similar, indicating that policy churn is unlikely strongly dependent on any of these specific choices.[4]

The effect also appears not specific to any environment: we measure similar magnitudes of policy change across a range of Atari games that vary in many dimensions, such as action space, reward scale and sparsity, deadliness, etc. Furthermore, it is present in all stages of training (see Figures 1 and 5). Unsurprisingly it it highest in early learning, but it remains high during training and even after evaluation performance converges or stabilises (possibly at the optimal level as in PONG).

3.2 Ablations

Redundant actions. A simple factor that could explain policy churn are redundant actions, i.e., when nominally different actions have the same outcome (generally, or in a large fraction of states). This property varies widely by environment (see also [31]), but we observe similar levels of policy change across different ones (Figures 1 and 5). Also, when exposing the full Atari action set in all games, creating explicit global redundancy, the churn magnitudes are not affected much (Figure 6). Appendix A.2 looks at the fine-grained aspect of *which* actions tend to be swapped for which others, and finds no structure easily related to the (known) equivalence relations (Figure 8). In other words, most $\arg\max$ changes are *not* happening between equivalent actions.

Small action gaps. Another potential factor for large policy churn with greedy policies can be the interplay between FA and small action gaps (difference between largest and second-largest action values): small approximation error can suffice for sub-optimal actions to overtake optimal ones. This hypothesis predicts that value learning methods inducing larger action gaps (e.g., Advantage Learning (AL, [3]) which artificially lowers the values of sub-optimal actions) could reduce policy churn. Figure 3 (right) shows that indeed policy churn is diminished substantially by AL, correlating with

[4]Given the high churn in (preliminary) experiments on DM-Lab [1], as shown in Figure 25, we also consider it unlikely that the phenomenon is specific to the Atari setting.

an increase of action gaps (which consistently grows under AL, see Figure 9). Curiously, this does not seem to severely diminish the remaining policy churn's effectiveness for exploration: Figure 10 shows successful training of an AL-DQN with $\epsilon = 0$.

Non-stationary state distribution. Another explanatory hypothesis for the observed large magnitude of policy churn relates it to the non-stationary data distribution caused by the evolving data generating policy. Even when the policy seems to have converged with near-optimal performance, like in PONG, policy churn may concentrate on states where action decisions are inconsequential (where multiple actions have near-equal value), causing non-stationarity in the data distribution and thereby driving further policy churn. To test this, we utilize the "forked tandem" setting from [34], in which a high-performing policy is trained on the stationary data distribution generated by its initial snapshot. As can be seen in Figures 3 (right) and 11, the high level of policy churn is still preserved in this stationary-data regime, ruling out data non-stationarity as the main driver of the phenomenon.

Non-stationary targets. Temporal difference (TD) learning can give rise to another form of non-stationarity, as the bootstrap targets change at the pace of the target value function, which is preserved even in the forked tandem setting. To sidestep this, a simple control experiment uses the same setup but with Monte-Carlo returns as learning targets, turning it into pure policy evaluation via supervised regression (with noisy targets). The results (Figures 3, right, and 12) show a similar level of churn to the Q-learning updates, indicating that the phenomenon is not specific to TD-based algorithms.

Decoupled acting and target networks. Using a purely greedy policy based on the agent's target network and varying the frequency with which it is synchronized to the online network allows us to assess how much policy churn is beneficial for exploration. To avoid conflation with an adverse effect on learning stability, we keep the regular target network update frequency constant, while using an additional *acting network*, periodically copied from the online network, for behaviour generation alone. Figure 13 shows that an acting network updated more than every $\approx 1\,000$ gradient updates achieves most of the exploration benefits of acting with the online network, though in some games higher frequencies yield further benefit. This implies that the amount of policy change needed for exploration is much smaller than what is generated by DQN's learning process (otherwise the observed collapse would happen at much smaller update intervals).

Self-correction and the tandem effect. In [34] a phenomenon dubbed the "tandem effect" was observed: the failure of a deep RL agent to adequately learn from the training data generated by a different instance of the same agent, highlighting the importance of *self-correction* by interactively generated data from the policy being trained. One of their settings, the "forked tandem", starts with a copy of a high-performing policy and uses data sampled from its stationary distribution to continue training; even this apparently benign scenario leads to instability and potential collapse of the trained policy. Policy churn may provide a partial mechanistic explanation for the origin of the instability, showing that rapid policy change can be expected at all stages of training. The observation that the trained policy changes on a significant proportion of states at every update (performed on a negligibly small sample, a single minibatch of 32 state transitions) supports the hypothesis from [34] that erroneous extrapolation or over-generalization may play a key role in causing deviation and instability and producing the tandem effect in the absence of corrective training signal from self-generated data. Analogously to results in [34] we observe that the magnitude of policy churn is highly correlated with the depth of the trained function approximator, further supporting this hypothesis; see Figures 3 (right) and 14, which show specifically how policy change drops dramatically in a linear FA regime.

3.3 Detailed case study: Catch

Catch is a toy environment where the total number of states is small enough to be amenable to ground-truth dynamic programming approaches using the explicit matrix of transition probabilities, while providing an observation space that requires non-linear FA. We construct a spectrum of settings that all learn the optimal policy after some P iterations. For each we measure cumulative policy

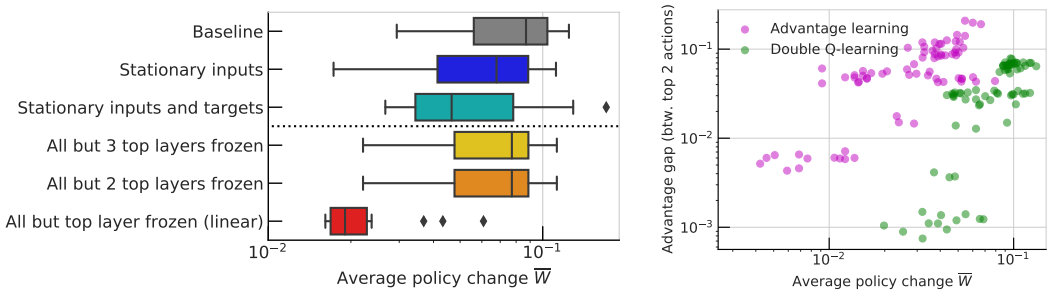

Figure 3: **Left:** Ablations in the "forked" DoubleDQN setting (5 games, 3 seeds each). Training progresses as normal until 50M frames, after which either the data distribution is fixed ("stationary"), or part of the neural network is frozen; "stationary targets" denotes regression onto Monte-Carlo returns. The only large effect on policy change is when switching from deep learning to linear FA (**red**). For more detail see Figures 11, 12, and 14. **Right:** Relation between policy change and advantage gaps. Different points correspond to different games, seeds and stages of learning. This shows that the action-gap-increasing algorithm (**purple**) reduces policy change; and also how, given an algorithm, advantage gaps correlate with churn. For more detail see Figures 9 and 10.

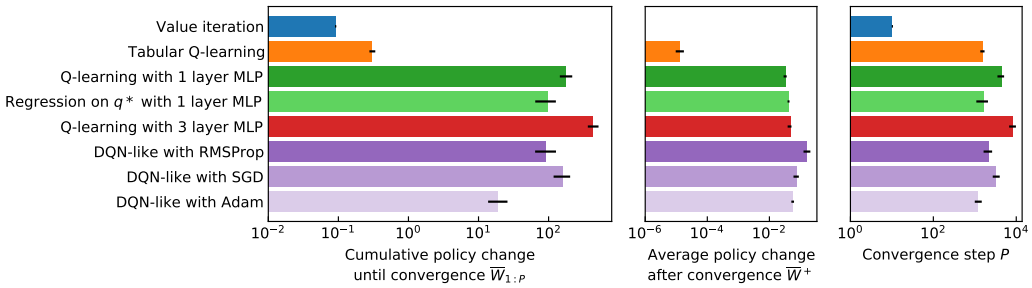

Figure 4: Ablations of what aspects drive policy change in Catch. Unless specified each FA variant uses the SGD optimiser. Here "DQN-like" is "Q-learning with 3 layer MLP" with replay and mini-batches. Error bars denote the interquartile range over 100 seeds.

change $\overline{W}_{0:P}$ and average change after convergence $\overline{W}^{+}$. Main results are shown in Figure 4. Sitting at one end of the spectrum, exact value iteration converges after $P = 10$ steps with $\overline{W}_{0:P} = 0.09$ (because the initial policy is random, and the optimal policy has ties in most states). The next simplest setting is tabular Q-learning with incremental updates (here and elsewhere, hyperparameters like the learning rate are tuned for fast convergence to optimal performance, see Appendix B.3 for details). The next steps add in non-linear FA (shallow, then deep). At the other end of the spectrum is an approximation to DQN which includes experience replay, mini-batches, a deep neural network, and an advanced optimiser (RMSProp [46]). Figure 4 also shows intermediate variations such as supervised regression to q^*, and different optimisers (e.g., Adam [23]); additional results in Appendix A.1.

3.4 Mechanistic hypothesis

In an attempt to synthesise all the evidence presented so far, we propose that the observed policy churn is primarily a result of two components that need to be present jointly:

1. Non-linear, global function approximation (such as deep neural networks), where each update can affect all states and all actions.
2. A learning process with a high amount of noise. This could have multiple sources: stochastic optimisation (e.g. small batch sizes, large learning rates), noisy learning targets, non-stationary data (or targets, as with bootstrapping), an imbalanced data distribution (e.g., seeing some actions far more often than others).

In other words, the rapid policy change dubbed churn is the symptom of high-variance updates to a global function approximator. A compounding effect could be a *mismatch* between the regression loss driving the learning process and policy space, which is what matters for performance and for exploration (and is measured here). While supervised learning does not have this mismatch, it could conceivably have large "policy change" too. An in-depth treatment exceeds the scope for this paper, but preliminary results on MNIST (Appendix B.5) indicate *no* large churn in the supervised setting.

4 Where do we go from here?

4.1 Learning at the edge of chaos

Deep learning has a well-known trade-off between speed of learning and stability that incentivises tuning the learning dynamics to be near the edge of chaos[5] [10]. The presence of policy churn could enrich this picture in two ways for the case of deep RL. First, to the extent that rapid policy change helps drive exploration (Section 2), this is an additional incentive to keep learning dynamics sufficiently noisy. Second, to the extent that self-correction is central to value-based learning [34], which relies on the actions to be corrected being picked by a greedy policy, this also encourages a large amount of policy change. Circumstantial evidence for these is that value-based RL tends to require additional stabilisation mechanisms (e.g., target networks), less common in policy-based RL.

4.2 Policy null-spaces

Policy change is necessary for policy improvement, but does not necessarily imply change in performance. We define the space of policies that have the same value (in all states) as a reference policy π as its *null space* $\mathrm{Null}(\pi)$. We can quantify its *diameter*

$$|\,\mathrm{Null}(\pi)| := \max_{\pi',\pi''\in\mathrm{Null}(\pi)} \overline{W}(\pi',\pi'')$$

as the largest policy change possible between any pair of policies within it. Typical scenarios with large null spaces have states in which actions have no effect (e.g., move actions while falling), multiple actions with the same effect (globally, or locally in part of state space), or multiple paths leading to the same outcome (e.g., up-then-left vs. left-then-up). These scenarios are common among the environments typically studied in deep RL. Appreciating that policy null spaces can be large makes the policy churn phenomenon more palatable, helping us reconcile that agents changing their mind a million times about the best action (on average in each state) can have excellent performance. A particularly interesting null space is the one around the optimal policy $\mathrm{Null}(\pi^*)$: when it is large, there are many "safe" ways to keep changing the policy (possibly by a lot) after converging to maximal performance, which is what we observe on, e.g., PONG (see Figure 7), at least when $\epsilon > 0$.

4.3 Off-policy corrections in the presence of churn

Most types of off-policy correction are based on the gap between the data-generating behaviour policy μ, and a target policy π. When doing multi-step updates, conservative methods truncate trajectories to bootstrap early compared to on-policy experience [25]. To obtain the benefit of multi-step value propagation, the average truncation length cannot be too short, and thus a silent assumption is that μ and π do not differ too frequently. Empirically however, multi-step back-ups without *any* off-policy correction can be surprisingly effective [20]. We can now re-interpret these findings through the lens of policy churn: as the greedy policy changes much faster than expected, even a slight latency (a few updates) between the parameters of the data-generating policy and the current target policy leads to massive truncation effect. If consecutive greedy policies are (approximately) within the null-space of each other, the benefits of an uncorrected multi-step update may outweigh its cost. It also suggests the possibility of new off-policy algorithms that exploit the knowledge of the churn phenomenon, by truncating in a less aggressive fashion, motivated by the intuition that most rapidly changing policies lie within an (approximate) null space of each other (see Appendix A.5).

[5] As epitomised in the heuristic to tune the learning rate to the largest value that does not explode.

4.4 The social dynamics of research

If policy churn is indeed a hidden form of exploration, how did it come about? It seems unlikely that a useful mechanism emerges completely by chance. An intriguing possibility is that policy churn is the effect of a gradual process of *natural selection* [12]. The hypothesis is that, if policy churn provides benefits, algorithms that display some level of it would be favoured over their counterparts in the inevitable engineering work surrounding the design of large-scale agents. In this view, RL practitioners play the role of "nature" exerting a selective pressure that shapes algorithms across time. This process could be completely *unconscious* to the researchers involved: agents under-performing due to weak exploration would be discarded in favour of agents that explored better using some form of (hidden) policy churn. Over time, the multiple degrees of freedom of large-scale agents (hyper-parameters, network architecture, etc.) would be tuned to reflect just the right amount of churn: enough to help in exploration, but not too much to make the overall learning process unstable.

As appealing as the above hypothesis may be, as of now we do not have any evidence to support it. Still, it is worth considering as it raises many interesting questions. Are there other hidden effects that have been selected for over the years? How many good design choices got discarded because they happened not to promote policy churn (or other similarly hidden effects)? Is the AI research community narrowly focused on a handful of design templates that happen to induce some ill-understood dynamics? This sort of question did not arise in the past, when agents were simple enough that we could keep track of their functioning at the finest level of detail. Now that deep RL agents have reached a certain level of complexity, any design choice may have a cascade effect whose consequences we do not anticipate. This creates the perfect environment for the sort of selective process described above. Acknowledging this possibility and being aware of it may be an important step to unveil hidden effects and perhaps turn them into more purposeful design.

5 Related work

A phenomenon in the literature that is related, but not equivalent, to policy churn is that of *policy oscillation* or *chattering* [16, 4, 48]. Bertsekas and Tsitsiklis [4] define the *greedy region* of a function $q \in \mathcal{Q}$ as the set of functions in $\mathcal{Q}$ that induce the same greedy policy as q. Each greedy region has an associated "fixed point": the value function of the greedy policy induced by the functions in it. Conversely, every value function is the fixed point of a greedy region. It is well known that the only value function that belongs to its own greedy region is the optimal value function q^* [4]. However, when function approximation is used, projecting value functions onto the FA space may create cycles that repeat indefinitely, a phenomenon known as policy oscillation. We do not think policy oscillation is a likely explanation for policy churn, because the approximators used in our experiments should have the capacity to approximate any value function to a reasonable level of accuracy, as indicated by the ablations in Figures 3 and 4 and Appendix A.1 that show neural networks with more or wider layers exhibit more churn. We believe it is more instructive to think of each update of the approximator as moving the value function across the boundaries of greedy regions. As discussed, in many cases this has no effect on the agent's performance, since the policies associated with neighbouring greedy regions may belong to each other's null space (Section 4.2).

Another body of related work is the literature on stochastic gradient descent, and how a relatively high amount of stochasticity can be beneficial to optimisation by overcoming local optima and converging to flatter optima with better generalisation properties [22, 37, 24]; to the extent that it may even be beneficial to inject additional noise with Langevin dynamics [50]. More loosely related are studies of learning in animals, exhibiting large drift in synaptic or representation space [30, 38, 26, 13, 42], as well as evidence for highly variable behaviour policies that can get consolidated through salient dopamine events [9].

6 Conclusions and Future Work

Revisiting interpretations. Nine years after the introduction of DQN [27], there are still phenomena in value-based deep RL that remain to be understood, with this paper putting the spotlight on one of them, which lies at the intersection of learning and exploration dynamics. In particular, we hope that an awareness of the churn phenomenon will make researchers revisit some good ideas that may have been prematurely disregarded or under-valued, either because their promised exploration effect was too entangled with learning dynamics and the resulting churn, or because they improved the stability of learning dynamics at the expense of reduced exploration that undid the overall gains.

Churn beyond value-based RL. An obvious follow-up question that we have not yet addressed is whether policy churn is also an important effect beyond value-based algorithms. We could imagine that actor-critic algorithms incur much less policy change, because stochastic policies change more smoothly, or because various penalty terms keep the updated policy from deviating too much from its precursor.[6] If this is indeed the case, that would indicate that exploration in these methods is different as well, possibly with complementary advantages and disadvantages. Similarly, various instances of model-based RL [29, 43] may have less policy change, because the planning process could mitigate some of the function approximation effects.

Explicit and controllable churn. If policy churn is indeed a valuable and non-trivial exploration mechanism, then it may be costly to deliberately abandon, especially if its effect turns out to be complementary to simpler noise-based exploration mechanisms. Ideally we would want an explicit and controllable mechanism that produces the same kind of consistent, non-harmful exploration behaviour, but without such a tight entanglement with the learning dynamics. The core benefit of such a division of labour would be that practitioners could study, change or tune the learning process for what is valuable to it, namely stability or representation learning; and separately adapt or tune the exploration process based on its differing criteria such as diversity, without having to trade one off against the other.

References

[1] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

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

[3] M. G. Bellemare, G. Ostrovski, A. Guez, P. Thomas, and R. Munos. Increasing the action gap: New operators for reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Feb. 2016.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[6] D. Budden, M. Hessel, I. Kemaev, S. Spencer, and F. Viola. Chex: Testing made fun, in JAX!, 2020.

[7] D. Budden, M. Hessel, J. Quan, S. Kapturowski, K. Baumli, S. Bhupatiraju, A. Guy, and M. King. RLax: Reinforcement Learning in JAX, 2020.

[8] A. Cassirer, G. Barth-Maron, T. Sottiaux, M. Kroiss, and E. Brevdo. Reverb: An efficient data storage and transport system for ML research, 2020.

[6] Preliminary results one one type of actor-critic agent indicate that policy churn is still large, see Figure 26.

[9] C. Clopath, L. Ziegler, E. Vasilaki, L. Büsing, and W. Gerstner. Tag-trigger-consolidation: a model of early and late long-term-potentiation and depression. *PLoS computational biology*, 4(12):e1000248, 2008.

[10] J. M. Cohen, S. Kaur, Y. Li, J. Z. Kolter, and A. Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

[11] W. Dabney, G. Ostrovski, and A. Barreto. Temporally-extended ϵ-greedy exploration. In *9th International Conference on Learning Representations (ICLR'21)*, 2021.

[12] C. Darwin. *The origin of species by means of natural selection*. John Murray, 1859.

[13] D. Deitch, A. Rubin, and Y. Ziv. Representational drift in the mouse visual cortex. *bioRxiv*, 2020.

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

[15] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration. *arXiv preprint arXiv:2107.02385*, 2017.

[16] G. J. Gordon. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, pages 261–268. Elsevier, 1995.

[17] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX, 2020.

[18] M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan. Optax: Composable gradient transformation and optimisation, in JAX!, 2020.

[19] M. Hessel, M. Kroiss, A. Clark, I. Kemaev, J. Quan, T. Keck, F. Viola, and H. van Hasselt. Podracer architectures for scalable reinforcement learning. *arXiv preprint arXiv:2104.06272*, 2021.

[20] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[21] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, and R. Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

[22] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] B. Kleinberg, Y. Li, and Y. Yuan. An alternative view: When does sgd escape local minima? In *International Conference on Machine Learning*, pages 2698–2707. PMLR, 2018.

[25] T. Kozuno, Y. Tang, M. Rowland, R. Munos, S. Kapturowski, W. Dabney, M. Valko, and D. Abel. Revisiting peng's q (λ) for modern reinforcement learning. In *International Conference on Machine Learning*, pages 5794–5804. PMLR, 2021.

[26] T. D. Marks and M. J. Goard. Stimulus-dependent representational drift in primary visual cortex. *bioRxiv*, 2020.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[29] T. M. Moerland, J. Broekens, and C. M. Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[30] G. Mongillo, S. Rumpel, and Y. Loewenstein. Intrinsic volatility of synaptic connections a challenge to the synaptic trace theory of memory. *Current Opinion in Neurobiology*, 46:7–13, 2017. Computational Neuroscience.

[31] M. J. Nelson. Estimates for the branching factors of Atari games. *arXiv preprint arXiv:2107.02385*, 2021.

[32] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped DQN. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[33] I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepesvari, S. Singh, et al. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*, 2019.

[34] G. Ostrovski, P. S. Castro, and W. Dabney. The difficulty of passive learning in deep reinforcement learning. *arXiv preprint arXiv:2110.14020*, 2021.

[35] M. Pîslar, D. Szepesvari, G. Ostrovski, D. Borsa, and T. Schaul. When should agents explore? In *International Conference on Learning Representations (ICLR)*, 2022.

[36] J. Quan and G. Ostrovski. DQN Zoo: Reference implementations of DQN-based agents, 2020.

[37] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[38] M. E. Rule and T. O'Leary. Self-healing neural codes. *bioRxiv*, 2021.

[39] T. Schaul, D. Borsa, D. Ding, D. Szepesvari, G. Ostrovski, W. Dabney, and S. Osindero. Adapting behaviour for learning progress. *arXiv preprint arXiv:1912.06910*, 2019.

[40] T. Schaul, G. Ostrovski, I. Kemaev, and D. Borsa. Return-based scaling: Yet another normalisation trick for deep RL. *arXiv preprint arXiv:2105.05347*, 2021.

[41] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

[42] C. E. Schoonover, S. N. Ohashi, R. Axel, and A. J. P. Fink. Representational drift in primary olfactory cortex. *Nature*, 594(7864):541–546, 2021.

[43] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

[45] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[46] T. Tieleman, G. Hinton, et al. RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[47] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[48] P. Wagner. Policy oscillation is overshooting. *Neural Networks*, 52:43–61, 2014.

[49] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, 2016.

[50] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

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 3 discusses the generality and applicability, Section 4 the ramifications.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 4.4 is specifically about the impact on the social dynamics of research.
   (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)? [No] Our implementation relies on some proprietary code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Many of our experiments have 3 seeds across 15 games which is enough to draw meaningful conclusions. We report either error bars, show box-plots, or show all seeds individually.
   (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 B.
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? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (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]