ZIPFIAN ENVIRONMENTS FOR REINFORCEMENT LEARNING

Stephanie C. Y. Chan*, Andrew Kyle Lampinen*, Pierre Richemond* & Felix Hill*
*Equal contributions
DeepMind
London
United Kingdom
{scychan,lampinen,richemond,felixhill}@deepmind.com

ABSTRACT

As humans and animals learn in the natural world, they encounter distributions of entities, situations and events that are far from uniform. Typically, a relatively small set of experiences are encountered frequently, while many important experiences occur only rarely. The highly-skewed, heavy-tailed nature of reality poses particular learning challenges that humans and animals have met by evolving specialised memory systems. By contrast, most popular RL environments and benchmarks involve approximately uniform variation of properties, objects, situations or tasks. How will RL algorithms perform in worlds (like ours) where the distribution of environment features is far less uniform? To explore this question, we develop three complementary RL environments where the agent's experience varies according to a Zipfian (discrete power law) distribution. These environments will be made available as an open source library.[1] On these benchmarks, we find that standard Deep RL architectures and algorithms acquire useful knowledge of common situations and tasks, but fail to adequately learn about rarer ones. To understand this failure better, we explore how different aspects of current approaches may be adjusted to help improve performance on rare events, and show that the RL objective function, the agent's memory system and self-supervised learning objectives can all influence an agent's ability to learn from uncommon experiences. Together, these results show that learning robustly from skewed experience is a critical challenge for applying Deep RL methods beyond simulations or laboratories, and our Zipfian environments provide a basis for measuring future progress towards this goal.

1 INTRODUCTION

Real world distributions are rarely uniform. For example, the words in natural language follow a power-law distribution known as Zipf's law (Zipf, 1936)—frequency is inversely proportional to rank. Thus common words like 'the' are extremely frequent, but there is a very long tail of rare words. Similar patterns hold in many other domains of human experience, such as the frequency of objects encountered in early visual learning (Clerkin et al., 2017; Smith et al., 2018) — see Figure 1 – or the relationships we form in social networks (Arenas et al., 2004; Albert & Barabasi, 2001). This aspect of natural experience poses a substantial challenge for statistical learning systems, since information about rare entities or events can get lost in a sea of frequent stimuli. Worse, in many domains, rare situations can be critically important. In language, rare words on average have a stronger influence on the meaning of sentences than frequent ones (Bybee & Thompson, 1997). Similarly, it may be less likely that a self-driving car observes a person rather than a lane marker in the road, but it is more critical to choose appropriate actions around the person.

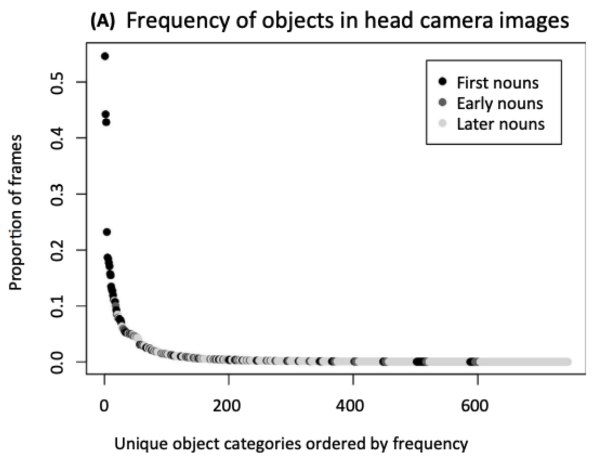


Figure 1: The highly-skewed frequency of common objects (cup, spoon, bowl), as observed in data recorded from 8-10 month old infants wearing head cameras. Reproduced with permission from Smith et al. (2018).

[1]Environments will be made available at https://github.com/deepmind/zipfian_environments

Reinforcement Learning (RL) is a powerful paradigm for developing artificial systems that can learn to take optimal decisions from continuous experience of an environment. However, simulated RL environments do not typically expose learners to data that follows a skewed, long-tailed frequency distribution, and as a consequence downplay the value of strong performance in rare situations. Environments that employ random programmatic variation typically vary uniformly (Chevalier-Boisvert et al., 2018; Cobbe et al., 2019; Zhang et al., 2018; Peng et al., 2018). In standard multi-task settings (Bellemare et al., 2013; Brockman et al., 2016), learners can automatically access equal amounts of experience for each of the multiple tasks. Even in hand-curated game environments, where the distribution of objects and properties is probably not uniform (e.g. Vinyals et al., 2017), it is unlikely that data fully reflects the sharp skew and heavy tails of natural experience. Moreover, irrespective of the distribution of experience from which agents learn, RL agents are typically evaluated under the same frequency distribution as experienced by the learner. Since the agent's score is averaged over the evaluation, performance in high frequency situations is credited most, and failure on rare situations can have little impact on evaluation metrics.

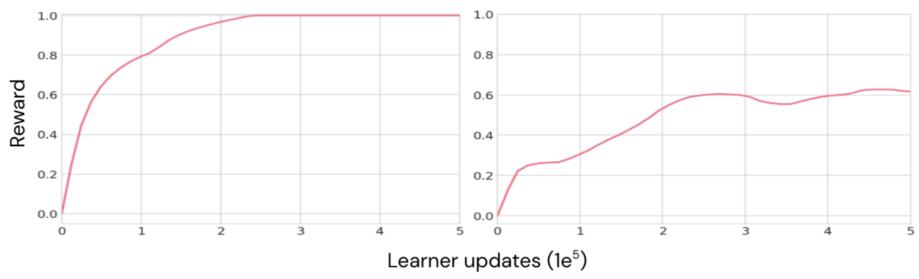


Figure 2: An agent trained with the IMPALA algorithm on in a Zipfian environment (Zipf's Playroom) with 50 objects. **Left** evaluated on all objects presented according to the training distribution **Right** on the rarest 20% of objects in the training distribution.

Figure 2 shows how extreme this effect can be: an agent is trained to respond appropriately to one of 50 possible objects presented according to a highly skewed distribution. When evaluated according to the training distribution, its performance looks almost perfect (left), but in fact when it encounters any of the rarest 10 objects, it behaves appropriately less than 60% of the time (right). Natural intelligence avoids similar failures, thanks to evolved mechanisms like the hippocampus, a rapid learning system that can store and recall individual experiences, complementing a second, slower system of parametric cortical learning (McClelland et al., 1995).

Can existing RL algorithms and popular agent architectures allow robust learning about both frequent and rare phenomena? Or will human-like systems for representation and memory be critical, as argued e.g. by Kumaran et al. (2016). To assist in resolving these questions, and to highlight the importance of developing systems that can learn from skewed or long-tailed distributions of experience, we develop three new benchmark environments. To parallel natural intelligence, two of these environments — *Zipf's Playroom* and *Zipf's Labyrinth* — place agents inside a 3D, first-person simulation. To enable faster experimentation at lower computational cost, the third environment — *Zipf's Gridworld* — is a top-down 2D world with a simple discrete action space in which the learner sees (and controls) its own avatar. In Zipf's Playroom, the agent must learn to identify and interact with a global set of 50 different objects which appear according to a heavily-skewed power-law (Zipfian) distribution. In Zipf's Labyrinth, the agent must learn to perform well across a set of 30 tasks when its experience across those tasks is again governed by a Zipfian distribution. Finally, in Zipf's Gridworld, both the agent's experience of both objects *and* spatial layouts are heavily-skewed.

Importantly, in all three environments, when evaluating, we measure agent performance under two new distributions. First, to quantify exclusively competence in rare situations, we compute average success when episodes are sampled uniformly across the rarest 20% of training situations. Second, to verify that performance on common situations is also preserved, we compute average success when episodes are sampled uniformly from *all* possible training situations.

It is worth noting that RL agents in any setting must learn from experience that is somewhat non-uniform. By their very nature, they must explore, and thus control and change the shape of the data-generating distribution they experience over time - the so-called 'generative model' form of exploration (Kearns et al., 1999). Intuitively, reinforcing states with high reward (exploitation) skews the state-visitation probability density, and existing RL algorithms are therefore already designed to handle some agent-induced non-uniformity. Furthermore, many recent RL innovations could potentially adjust for non-uniformity, such as upweighting data that most 'surprises' the learner. In particular, Prioritized Experience Replay (PER; Schaul et al., 2016) observes that the empirical distribution of TD-errors in the experience replay buffer of Atari games is heavy-tailed and follows a power law; and that emphasizing surprising transitions promotes better learning. Thus PER could potentially address some of the skew induced by the environment.

However, while these existing methods can learn despite the inevitable *agent-driven* non-uniformity of experience, their effectiveness in the face of a *environment-driven* non-uniformity of experience has not been explored. Indeed, it is likely that these two sources of skewed experience interact, and that experience in a highly-skewed environment may actually exacerbate agent-induced non-uniformity of experience. For example, if a particular goal is more frequently rewarding than others, the agent may further bias its experience by consistently pursuing that goal, and thus potentially fail to learn under what conditions the other goals are rewarding.

In an initial exploration of these questions, we evaluate a range of popular RL algorithms and agent architectures on our three Zipfian benchmarks. We focus particularly on three aspects of agent design that may influence performance in rare situations: the *learning objective* (or RL loss), the *memory architecture*, and applications of *self-supervised learning*. We find that, despite apparently 'mastering' their training tasks, current popular algorithms in fact perform poorly in rare situations – in many cases barely above chance. Modified learning objectives, memory architectures and self-supervised auxiliary learning can each lead to moderate improvements in rare situations, but performance is still substantially lower than that achieved when training experience is uniform rather than skewed. Our findings pose an important challenge for future research, which is exemplified in our three new benchmarks – to create learning systems and RL algorithms that can leverage the plentiful access to frequent experiences provided by natural data and real-world learning contexts to perform successfully, robustly, and safely when faced with rare situations.

2 Benchmarks for learning from skewed experience

We develop three benchmark environments in which the training environment layouts, objects, or tasks are experienced according to a Zipfian or power law distribution (Newman, 2004) with exponent $\alpha \in [0, \infty)$. That is, the probability distribution for a random variable X is:

$$p(X = x) = \frac{1}{Z} \cdot \frac{1}{x^{\alpha}} \tag{1}$$

where Z is a normalizing constant. α thus determines the degree to which the agent's experience is skewed and long-tailed. A higher value of α increases the difference in frequency between common and rare events, and thus should make learning about rare situations more difficult. Note that a larger support for X will also increase the skew.

The particular values of α chosen for each environment were intended to create a difficult but not insurmountable learning challenge. However, in both Zipf's Labyrinth and Gridworld it is possible to change the training distributions quite easily; thus researchers could explore different settings when using the environments. The chosen skew levels are within range of many real-world skew levels (e.g. Newman, 2004; Piantadosi, 2014).

The three benchmark environments we propose cover a range of settings and paradigms—including 2D and 3D environments, language conditioned or visually-cued tasks, etc. Furthermore, these environments target a diverse range of behaviors, ranging from navigation to instruction-following and

2.1 Zipf's Playroom

Zipf's Playroom (Figure 3) is a 3D room built using the Unity game engine. This environment focuses on a highly skewed experience of objects (and their corresponding labels), which parallels the experience of human infants and their highly non-uniform experience of objects and people (Clerkin et al., 2017; Smith et al., 2018). We consider two tasks: a lifting task and a putting task. In the **lifting task**, the agent is placed in a room containing three objects. The agent receives a string instruction to lift a specific object. If the agent lifts the correct object, it receives a reward of 1 and the episode terminates. If the agent lifts an incorrect object, it receives a reward of 0 and the episode terminates. In the **putting task**, the agent is placed in a room containing three objects, a red box and a blue box. The agent receives a string instruction to pick up a specific object and to put it into one of the boxes. If the agent puts the correct object into the correct box, it receives a reward of 1 and the episode terminates. If the agent puts the correct object into the incorrect box or an incorrect object into either box, it receives a reward of 0 and the episode terminates.

In both tasks, we sample the three objects that appear in each episode according to a distribution over a total set of 50 objects. During training, these are sampled (independently, without replacement) according to the Zipfian distribution 1 (with exponent $\alpha = 3$ for lifting, and $\alpha = 2$ for putting). However, we evaluate the agent's behaviour when they are sampled from a uniform distribution (**uniform evaluation**), or from a uniform distribution truncated to the 10 least frequent items in the Zipfian training distribution (**rare item evaluation**). Once the three objects are determined, the target object (and hence the instruction) is determined according to a uniform choice between the three sampled objects. In all cases, evaluation consists of running 1000 episodes on the environment tasks, to estimate expected performance under the joint distribution of task-orthogonal and object-sampling randomness.

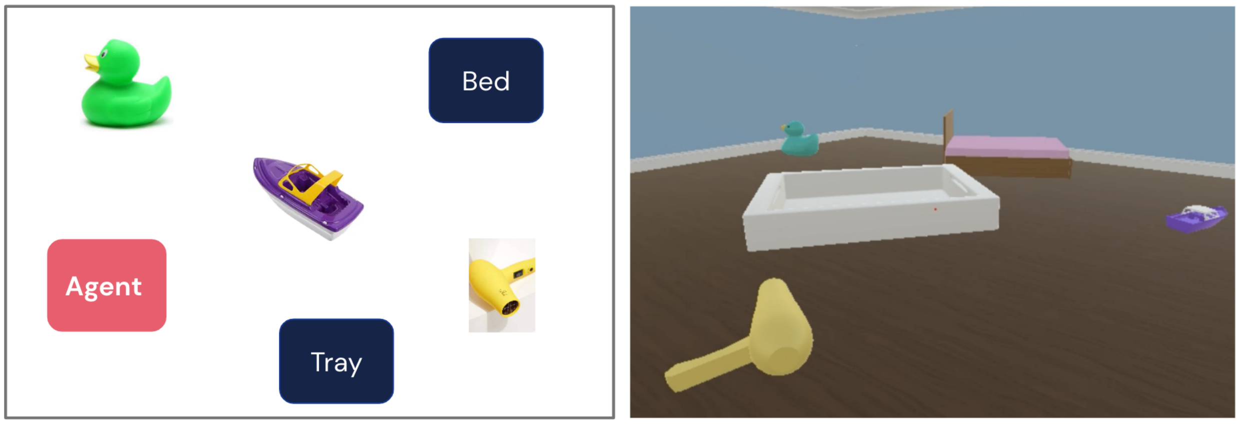


Figure 3: Illustration of Zipf's Playroom. **Left** a schematic of the room from above, showing three randomly chosen toys and the bed and tray furniture. The initial position of each entity and the agent is randomized per episode. The agent would receive an instruction like "Put a duck on a bed." **Right** A frame of the same environment (with different randomized positions) from the agent's perspective.

In training and testing, we vary several task-orthogonal aspects of the environment randomly according to a uniform (continuous or categorical) distribution: the initial positions and orientations of all objects, boxes and the agent. The variation during training encourages the agent to learn more general policies (Cobbe et al., 2019), while the variation at evaluation reduces the possibility that observed performance is due to some idiosyncratic bias.

The putting task additionally allows us to investigate the interaction between long-tailed distributions and systematic or compositional generalization (Fodor & Pylyshyn, 1988; Lake & Baroni, 2018; Hill et al., 2019). Training is performed on a curricula of three tasks that are trained on simultaneously. In order of increasing difficulty: lifting (same as above), putting near (same as the full task except reward is also given if the correct object is placed near the target box), and putting on (the full task). For 'putting near' and 'putting on', we only trained on the most frequent 20% of the 50 objects. Thus, to perform well at evaluation on the non-frequent objects, the agent must perform systematic generalization by composing its knowledge of the non-frequent objects with its knowledge of putting (combinations that were never required in training) – this is an especially difficult setup for systematic generalization, because of diminished experience with the non-frequent objects.

2.2 Zipf's Labyrinth

While Zipf's Playroom focuses on object identification and manipulation, *Zipf's Labyrinth* (Figure 4) focuses on heavily-skewed experience of tasks and situations that pertain to specific goals. To create it, we simply re-balance the existing *DM-Lab* benchmark (Beattie et al., 2016) – a collection of 30 distinct tasks set in a 3D, first person environment built on Quake 3 Arena.[2] During training, agents experience each task with probability determined by a Zipfian distribution (with exponent 1). We create this Zipfian distribution according to the original ordering of the tasks, which groups the tasks by type. This may amplify the challenge caused by the skew, because entire clusters of similar tasks will be rare or common. To ensure that our results are not solely determined by this particular ordering, we trained both on Zipfian distributions that decreased across the tasks ("forward Zipf") and also that increased across the tasks ("reversed Zipf"). As above, performance is measured across all tasks sampled uniformly, and also on the rarest 20% (i.e. six tasks).

2.3 Zipf's Gridworld

Zipf's Gridworld (Fig. 5) is a lighter-weight set of tasks for easier experimentation. To make experimenting more accessible, this environment has a simpler action space, shorter episodes, and only a visual input modality. The tasks are visually-cued object finding—the agent has to find the object depicted in the top-left corner of its visual input—in a 2D map containing several rooms and many objects. There are a fixed set of 20 maps, each of which has 9 rooms and 20 objects. The agent start location, as well as the object shapes, colors, and locations are fixed within each map. On each episode, the agent is visually cued to go to a particular target object (see Figure), and rewarded if it

[2] https://github.com/id-Software/Quake-III-Arena

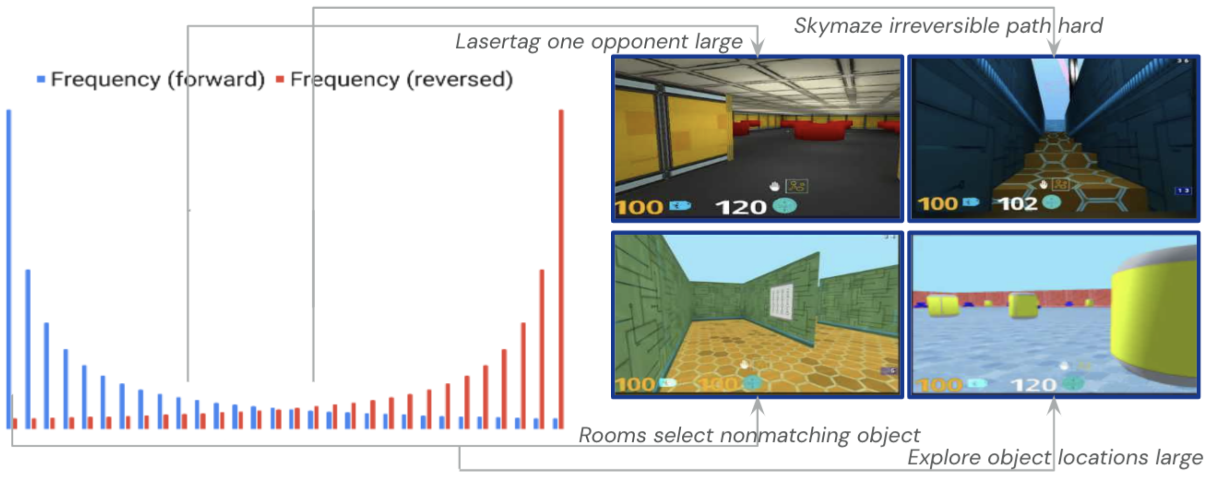


Figure 4: **Left** the frequency with which the learner experiences each of the DM-Lab 30 tasks in Zipf's Labyrinth. **Right** Frames of several tasks from the agent's perspective.

moves to that object (the episode ends with no reward if it touches any other object). Because the locations remain consistent within each map, it suffices for the agent to memorize the sequence of actions leading from each map's starting location to each goal object. However, the agent encounters the maps with a Zipfian distribution (with $\alpha = 2$), so that some are much more common than others, and within each map the goal is also chosen according to a Zipfian distribution. Zipf's Gridworld thus evaluates the agents ability to generalize from hierarchically skewed experience to uniform distributions over maps and goal objects, as well as uniform distributions over the rarest 20% of each—the 4 rarest objects on the 4 rarest maps.

3 Reinforcement Learning from rare experiences

Various aspects of popular (Deep) RL agents could influence their ability to learn effectively from rare experiences.

Prioritized experience replay. Prioritized experience replay (PER; Schaul et al., 2016) is designed to help replay-buffer-based Deep RL algorithms (Lin, 1992; Mnih et al., 2015) by prioritizing 'surprising' transitions. Each transition is assigned a priority based on the magnitude of its temporal-difference (TD) error, which is an observable (but imperfect) surrogate for its contribution to learning. Motivated by the empirical finding that the distribution of TD-errors stored in Atari experience buffers approximately follows a power-law distribution, the PER correction samples transitions proportional to a power law over the priority. The ablation study in Hessel et al. (2018) found that PER forms a crucial component of the performance of integrated Deep RL agents. The performance gains provided by similar rebalancing strategies are obvious in supervised learning as well (e.g. Hinton, 2007). Thus, PER's emphasis on surprising experiences, may help compensate for environment-induced skew in experience by increasing the priority of rare events. However, PER has not been tested specifically for its effect in skewed environments. This motivates our empirical analysis of the benefit of PER for multi-task settings in Zipfian worlds.

Self-supervised learning. Self-supervised learning (SSL) (Pathak et al., 2016; Doersch & Zisserman, 2017; Oord et al., 2018; Hénaff et al., 2020; He et al., 2020; Chen et al., 2020; Grill et al., 2020; Mitrovic et al., 2021; Caron et al., 2021; Radford et al., 2021) is a family of methods that aim to leverage the inherent structure of the training data in order to learn without supervision. In theory, many SSL methods should lead to grouping similar data points together somewhat irrespectively of the frequency with which those points appear; in fact some self-supervised learning algorithms were derived from (or admit) a *clustering* interpretation (Caron et al., 2020; Asano et al., 2020; HaoChen et al., 2021). Indeed, there is some empirical evidence that SSL can make supervised learners more robust to dataset imbalance (Liu et al., 2021), and in particular Zipfian shaped or long-tailed distributions of input data (Zhang et al., 2021b). Similarly, difficult and risk-sensitive problems in real world computer vision settings featuring a long tail of open classes such as medical (Ghesu et al., 2022) or autonomous driving data (Mittal et al., 2020) have been addressed using SSL.

However, the theoretical motivations for self-supervised reconstruction losses in RL are ambiguous, due to potential misalignment between the geometry of representations and environment structure (Zhang et al., 2021a). Tackling this misalignment may require more sophisticated representation-learning approaches (ibid); alternatively, SSL methods

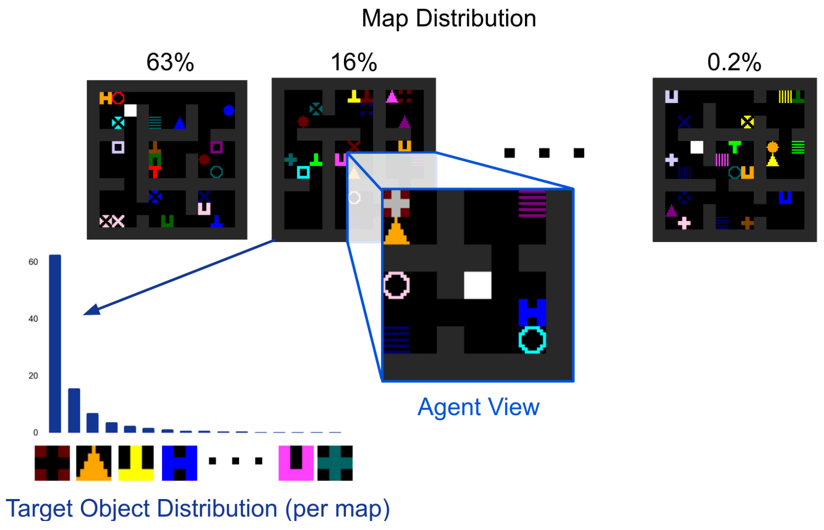


Figure 5: **Zipf's Gridworld: a 2D world with hierarchical skew.** In each episode, a map is picked from a Zipf distribution over 20 possible maps (top). The agent (white square) and objects always start at the same location within a given map. Each map contains 20 objects, and one of them is chosen to be the target according to a Zipf distribution (bottom left). Thus there is skew across the maps, and skew within each map. The agent sees a local region (bottom right), and the target object is displayed in the top left corner of its view over a light grey highlight background (here, the red collection of squares on the light background is the target object cue).

may require careful tuning to each environment and task, to accommodate differences in observation-task alignment. Nevertheless, previous work has shown empirically that SSL can improve the robustness and sample-efficiency of RL (Jaderberg et al., 2016; Nair et al., 2018; Guo et al., 2018; Wu et al., 2019; Guo et al., 2020; Schwarzer et al., 2021; Ye et al., 2021). Thus we evaluate whether SSL can improve robustness of learning in Zipfian environments.

Memory systems. It is also possible that the agent's within-episode memory architecture affects its ability to learn from rare events. In particular, transformers (Vaswani et al., 2017) have shown substantial ability to learn about rare events, with the largest language models exhibiting recall of some rare training experiences (e.g. Carlini et al., 2022). While evaluating such a large transformer would be prohibitively difficult, we compare to agents with a smaller transformer-style memory, to evaluate whether the transformer architecture can affect learning from rare experiences. Specifically, we compare to a Gated TransformerXL memory, which has been previously shown more rapid and stable learning in RL than non-gated transformers (Parisotto et al., 2020).

4 Experiments

We experiment with three popular RL algorithms that exhibit strong performance in 3D first-person environments. Two are policy gradient algorithms — IMPALA (Espeholt et al., 2018) and V-MPO (Song et al., 2020) — and one is a recurrent, distributed version of deep Q-learning (Mnih et al., 2015) — R2D2 (Kapturowski et al., 2018). In each case, the agent architecture consists of an *encoder*, for embedding visual (pixel) and language (string) observations from the environment engine, a *memory core* for integrating these observation embeddings over time, and policy, value or Q-value *heads* for mapping the output of the memory core into a representation that can inform behaviour. The encoder consists of a small convolutional network (Fukushima, 1980) and LSTM networks whose sizes depends on the environment. By default, the memory core is an LSTM (Hochreiter & Schmidhuber, 1997), but, we also explore the effect of a Transformer-based memory (Parisotto et al., 2020).

To explore the effect of SSL, we employ a generic reconstruction loss (as per Hill et al., 2020), in which additional networks are trained, conditioned on the state h_t of the agent's memory, to predict either the current visual (o_t - containing pixel values) or language (l_t) observation. This prediction network is either a deconvolutional network or a sequential (LSTM) language decoder, and the prediction loss (a cross-entropy loss, in the case of vision, and a sequence likelihood loss, for language) is backpropagated into the agent's core together with the RL (policy-gradient) loss. We also experiment with a predictive SSL loss that adapts the BYOL algorithm (Grill et al., 2020) to an RL setting.

Finally, we investigate the benefit of prioritized experience replay in the context of the replay-based R2D2 agent.

		Lifting task		Putting task	
		All items (uniform)	Rare items	All items (uniform)	Rare items
RL algorithm	IMPALA	0.65 ± 0.02	0.59 ± 0.02	0.62 ± 0.02	0.49 ± 0.04
	V-MPO	0.39 ± 0.01	0.40 ± 0.03	0.00 ± 0.00	0.00 ± 0.00
	R2D2	0.57 ± 0.00	0.51 ± 0.00	0.61 ± 0.05	0.44 ± 0.08
SSL (IMPALA +..)	Vis recon	0.66 ± 0.02	0.62 ± 0.06	0.67 ± 0.05	0.55 ± 0.10
	Vis+lang recon	0.64 ± 0.01	0.66 ± 0.04	0.68 ± 0.01	0.58 ± 0.04
	BYOL	0.55 ± 0.01	0.53 ± 0.03	0.62 ± 0.25	0.50 ± 0.18
Memory system (IMPALA +..)	MLP	0.47 ± 0.01	0.45 ± 0.02	0.00 ± 0.01	0.00 ± 0.01
	LSTM	0.65 ± 0.02	0.59 ± 0.02	0.62 ± 0.02	0.49 ± 0.04
	Transformer	0.46 ± 0.05	0.46 ± 0.04	0.59 ± 0.30	0.43 ± 0.24
	Trnsf.+vis+lang recon	0.57 ± 0.00	0.52 ± 0.01	0.50 ± 0.10	0.36 ± 0.08

Table 1: Evaluation performance in **Zipf's Playroom**, and the effects of different RL algorithms, self-supervised learning objectives and memory systems. We report median performance across a large evaluation window ([300k, 400k] updates) for the median of three runs ($\pm$ median absolute deviation across runs). Train performance (assessed on the Zipfian distribution) was generally much higher than evaluation performance, at 0.97-1.00 for lifting and 0.97-0.99 for putting (excluding V-MPO and IMPALA+MLP)—see Appendix B.1 for training results.

For each condition, we set hyperparameters by manual (informal) search guided only by performance on the skewed training set, not the evaluation environments. We initialized these searches with hyperparameters that have been shown to work well in uniformly-distributed versions of environments similar to ours. When we have found hyperparameters that enable strong performance on the training set, we fix them and train three randomly-initialised agent replicas per condition to convergence. We take the median run across the three runs, and report the median performance in a time window that covers the relatively converged sections of the learning curves. The windows we used were (number of learner updates): [300k, 400k] for Zipf's Playroom, [200k, 380k] for Zipf's Labyrinth, and [2M, 3M] for Zipf's Gridworld. However, evaluating in other similar windows would not have substantially altered the results in most cases, at least in terms of the overall ordering of performance, although performance might continue to slowly improve with longer training. For the Zipf's Labyrinth results, we normalized all results to human-level performance (as reported in Espeholt et al. 2018) to better aggregate performance measures across tasks, which can have very different ranges of reward values. All hyperparameters, together with other details of the agent networks are reported in Appendix A.5.

5 Results

Our results are summarized in Table 1 for Zipf's Playroom, Table 2 for Zipf's Labyrinth, and Table 3 for Zipf's Gridworld. We report training performance for each experiment in Appendix B.1. Full learning curves are provided in Appendix C.

Across all environments, we see that evaluation performance on rare scenarios is consistently lower than in more frequent situations, sometimes dramatically so. This difference is obscured when we evaluate on the (Zipfian) training distribution, and even (to a lesser extent) when evaluating uniformly across all items, tasks, or maps. Beyond reporting these basic differences in performance in rare situations, our experiments were designed to evaluate whether different architectural and algorithmic components can ameliorate these differences.

Does prioritized experience replay improve performance in rare situations? While on some tasks, training performance of the R2D2 algorithm was lower than our baseline IMPALA agent, we did find that PER made a difference to the underlying Q-learning approach on rare items. R2D2 without prioritized experience replay performed 17% lower on Zipf's Playroom putting tasks (0.51 average on all items, 0.36 on rare items) than the same algorithm using PER. We tried increasing the prioritization exponent in an effort to handle the more extreme skew in our environments, but we found it made little difference.

Does self-supervised learning improve performance in rare situations? Comparing the baseline IMPALA agent against IMPALA combined with various SSL algorithms, we see some moderate signs that self-supervised learning could improve agent performance in rare situations. In Zipf's Playroom, adding a vision+language based reconstruction loss to the IMPALA policy gradient loss improves performance by $+12\%$ on rare items in the lifting task and $+18\%$ on the putting task, with vision and language contributing approximately equal amounts to this uplift. These improvements are beyond the benefits of SSL more generally, vs. for the rare items specifically, e.g. as measured by the change in uniform-all performance (-1% for lifting and $+10\%$ for putting). In Zipf's Gridworld, where lan-

		Forward Zipf			Reversed Zipf		
		All tasks (Zipfian)	All tasks (uniform)	Rare tasks	All tasks (Zipfian)	All tasks (uniform)	Rare tasks
RL	IMPALA	0.55 ±0.26	0.31 ±0.09	0.32 ±0.11	0.45 ±0.10	0.31 ±0.04	0.35 ±0.05
	R2D2	0.46 ±0.25	0.26 ±0.06	0.30 ±0.11	0.47 ±0.16	0.31 ±0.06	0.36 ±0.14
SSL (IMPALA +..)	Vis recon	0.57 ±0.22	0.36 ±0.05	0.35 ±0.07	0.46 ±0.09	0.31 ±0.04	0.34 ±0.07
	Vis+lang recon	0.58 ±0.25	0.37 ±0.07	0.34 ±0.09	0.45 ±0.13	0.30 ±0.06	0.36 ±0.13
	BYOL	0.53 ±0.26	0.29 ±0.08	0.27 ±0.10	0.05 ±0.01	0.05 ±0.01	0.07 ±0.03
Memory system (IMPALA +..)	MLP	0.40 ±0.24	0.17 ±0.05	0.14 ±0.05	0.04 ±0.01	0.05 ±0.01	0.07 ±0.03
	LSTM	0.55 ±0.26	0.31 ±0.09	0.32 ±0.11	0.45 ±0.10	0.31 ±0.04	0.35 ±0.05
	Transformer	0.46 ±0.25	0.19 ±0.06	0.05 ±0.01	0.35 ±0.11	0.22 ±0.05	0.22 ±0.06
	Trnsf.+vis+lang recon	0.55 ±0.26	0.31 ±0.09	0.29 ±0.10	0.37 ±0.11	0.28 ±0.06	0.36 ±0.16

Table 2: Evaluation performance in **Zipf's Labyrinth**, and the effects of different RL algorithms, self-supervised learning objectives and memory systems. We report median performance across a large evaluation window ([200k, 380k] updates) for the median of three runs, and an average or weighted average across tasks (± standard error across tasks). Training performance was consistently higher, as shown in the column "All tasks (Zipfian)", compared to evaluation uniformly across all tasks, or on rare tasks only.

		All maps and objects (uniform)	Rare maps and objects
RL	IMPALA	0.69 ±0.05	0.24 ±0.04
	R2D2	0.35 ±0.06	0.18 ±0.01
SSL	Vis recon	0.80 ±0.03	0.29 ±0.07
	BYOL	0.31 ±0.02	0.03 ±0.01
Memory system (IMPALA +..)	MLP	0.77 ±0.01	0.13 ±0.06
	LSTM	0.69 ±0.05	0.24 ±0.04
	Transformer	0.82 ±0.01	0.22 ±0.02
	Transformer+vis+lang recon	0.70 ±0.01	0.23 ±0.04

Table 3: Evaluation performance in **Zipf's Gridworld**, and the effects of different RL algorithms, self-supervised learning objectives and memory systems. We report median performance across a large evaluation window ([2M, 3M] updates) for the median of three runs (± median absolute deviation across runs). Train performance (assessed on a Zipfian distribution) was generally much higher than evaluation performance, at 0.85 for IMPALA and 0.99 for IMPALA + Vis recon—see Appendix B.1 for detailed training results.

guage is not part of the environment, the equivalent visual SSL yields a $+20\%$ improvement on the rarest situations (and $+16\%$ when evaluated uniformly across all maps and objects. It is notable that both Zipf's Playroom and Zipf's Gridworld emphasise the agent's ability to distinguish objects of different categories, which is comparatively less important in Zipf's Labyrinth. This may explain why the benefit of SSL was lower there ($+3-6\%$). Note also that our approach based on BYOL failed, possibly because for simplicity, we did not enforce temporal consistency of our representations, or specifically tune an augmentations set to the problem.

Do memory architectures improve performance in rare situations? While transformer architectures have enabled large language models to improve performance on rare experiences (Carlini et al., 2022), we saw little evidence that replacing the agent's core memory with a transformer (instead of an LSTM or MLP) led to better performance on rare items. There are many differences in the experience, objective, and scale that could potentially explain this difference—for example modern language models can condition on words across many consecutive sentences, while the IMPALA and V-MPO algorithms do not enable an agent to condition on stimuli outside of the current episode. This motivates research into systems that allow agents to take decisions based on (memories of) extra-episodic as well as episodic context. We also did not find

Do RL agents or supervised learners perform better on rare stimuli? As noted previously, RL agents must contend with both agent-induced and environment-induced non-uniformity when learning from their experience. In contrast, supervised learners only contend with environment- (or data-) induced non-uniformity. We might then expect that — everything else being equal — a supervised learning system would perform better in rare situations than an RL agent. To test this hypothesis, we created a bespoke 'finding' task in Zipf's Playroom. In this task, the agent spawns facing two objects, one on the left and one on the right, randomly chosen according to a Zipfian distribution as before. We then trained an RL agent to approach a specific object given its name. At the same time, we trained a supervised classifier with the same encoders as the IMPALA agent to predict either *left* or *right* given the first-frame of an episode,

	All items (Zipfian)	All items (uniform)	Rare items
RL	1.00 ± 0.00	0.78 ± 0.01	0.66 ± 0.02
Supervised classifier	1.00 ± 0.00	0.79 ± 0.00	0.64 ± 0.02

Table 4: Comparison between an RL agent (IMPALA + LSTM) and a maximum-likelihood supervised classifier, on an object identification 'finding' task in Zipf's Playroom. We report median performance across a large evaluation window ([2M, 3M] updates) for the median of three runs (± median absolute deviation across runs).

via a cross-entropy loss. At test time, we evaluated the RL agent and the supervised learner, as before, according to uniform distributions on all items and on rare items. If, by chance, both objects in the room were the same, either outputs *left* or *right* were credited it as a correct prediction (as would be the case for the RL agent).

Contrary to our expectations, we found that the RL agent and supervised classifier performed almost identically (Table 4). In fact, if the learning curves are plotted as a function of the number of learner updates, the median learning curves are very similar. One possible explanation for this is that, while the reinforcement learner has to contend with agent-induced non-uniformity of experience, its embodiment also affords it access to much more data about rare items (in the form of views from different distances and angles). As has been discussed in prior work (Smith & Slone, 2017; Hill et al., 2019), this data might afford RL agents with uniquely rich representations which, in this case, may make up for the greater non-uniformity faced by the RL agent.

We note that our results only show that existing RL methods cannot *consistently* handle long-tailed distributions; they do not show that these methods can never handle long-tailed distributions at all. We also note that, given the statistical power of the experiments, we cannot over-interpret the differences between RL baselines; the goal here was not to give precise measures for each of these baselines.

5.1 Future directions

These empirical observations motivate several possible directions for future research.

Reweighting: While reweighting may be part of a general solution to the problem of learning from heavily-skewed experience, such a solution should not rely on knowing in what ways the distribution is skewed—in real-world applications, it might be difficult to infer. For example, manually reweighting the tasks from Zipf's Labyrinth would not be a satisfying solution to our benchmark. Instead, reweighting schemes should rely on statistics that the agent can observe—such as TD-error—or on clustering based on the agent's representations.

Distributional RL: Another avenue worthy of exploration is *distributional* reinforcement learning (Bellemare et al., 2017; Dabney et al., 2018a;b). Because such algorithms learn a full distribution of state-value functions rather than a single scalar, they may be more suited to handling extremely rare events and rewards that would otherwise be subsumed by the computation of a single expectation.

Exploration strategies: One role of explicit exploration strategies—and notions such as *intrinsic motivation*—is to counterbalance non-uniformity of agent experience by visiting states as uniformly as possible. In a bandit or tabular setting, provably efficient exploration strategies exist (Brafman & Tennenholtz, 2002; Lattimore & Hutter, 2014; Azar et al., 2017), usually relying on the principle of optimism in the face of uncertainty. In the neural network approximation case, explicit state visitation counts are unavailable, and proxies or bonuses (Bellemare et al., 2016; Ostrovski et al., 2017; Burda et al., 2019) must be employed. In an environment in which some objects or situations are much more frequent than others, exploration strategies might encourage agents to prefer those that are less frequent, and could thus play a role in improved performance on our Zipfian benchmarks. At the opposite end of the exploration spectrum (when it is not possible), distributional shift and generalization to unfamiliar states are also particularly relevant issues in *offline RL* (Levine et al., 2020).

Explicit recall of episodic memories: Kumaran et al. (2016) argue that natural intelligence relies on complementary learning systems, combining slower weight-based learning with episodic memory of experiences in the hippocampus; Kumaran et al. emphasize that these complementary systems "may allow the general statistics of the environment to be circumvented by reweighting experiences," thus allowing intelligent agents to learn rare-but-important things. Specifically, episodic memories can either allow preferential replay of important memories (as we explored with PER), or allow explicit recall of the memory when it is relevant. While a variety of memories have been proposed that allow RL agents to explicitly recall past experiences, these mostly are applied within a single episode (e.g. Wayne et al., 2018; Parisotto et al., 2020), or across at most a small number of tasks or episodes (Ritter et al., 2020; Lampinen et al., 2021). It is unclear whether these methods could be scaled to an entire lifetime of memory—as would be needed to handle Zipfian skew or other similar data distributions. Thus biologically motivated alternatives capable of explicitly recalling specific events at evaluation time, such as episodic control (Lengyel & Dayan, 2007; Blundell et al., 2016; Pritzel et al., 2017), may provide an exciting direction for future investigation.

6 Related Work

The interplay of rare events and RL, the explicit focus of the present work, was previously studied in the tabular setting with linear evaluation in Frank et al. (2008), where importance sampling methods were devised to reduce variance. To our knowledge, however, no works directly investigate performance on rare states for Deep RL algorithms trained on an explicitly Zipfian distribution of states or environments.

A number of studies in computer vision have investigated the challenge of learning from skewed training data in supervised settings, e.g. on objects (Zhu et al., 2014; Clerkin et al., 2017) or faces (Liu et al., 2015). This challenge has been formalised in *Long-tailed ImageNet*, a computer vision benchmark in which training data is heavily-skewed (Liu et al., 2019). Our Zipfian environments provide an analogous test for embodied reinforcement learning agents.

In such supervised settings, it is common to simply re-sample or re-weight classes so that the model is exposed equally to all classes Van Hulse et al. (2007); some more sophisticated methods e.g. produce artificial minority samples (Chawla et al., 2002). For evaluation, metrics such as balanced accuracy or the F-measure are commonly used to account for imbalanced classes (Johnson & Khoshgoftaar, 2019). However, these metrics and reweighting techniques are not applicable when the non-uniformity is hidden within classes—a common phenomenon known as 'hidden stratification'—which can have serious impacts on real-world applications such as medical imaging (Oakden-Rayner et al., 2019), and which some works have attempted to address (Sohoni et al., 2020; Sagawa et al., 2020).

In RL, by contrast, agents are typically evaluated on the same problem they are trained on. Therefore achieving reasonable performance on evaluations that differ from the training distribution (as in our proposed benchmarks) remains an open and challenging problem (Zhang et al., 2018; Hessel et al., 2019; Cobbe et al., 2019; Kirk et al., 2021). For instance, (Cobbe et al., 2019) find that training on dozens of thousands of levels is required to bridge the generalization gap in their multi-task setting, even when using just a uniform distribution of levels; our Zipfian case is likely to exacerbate the issue. The matter is also complicated by concerns around statistical significance of evaluation (Chan et al., 2020).

7 Conclusions

The structure of the real world is not uniform. Instead, the world is full of Zipfian distributions in which some entities, objects, or tasks are very common, but most are rare. We have highlighted the challenge this poses for reinforcement learning, and have proposed three diverse benchmark environments in which to investigate various aspects of this challenge—Zipf's Playroom, Zipf's Labyrinth, and Zipf's Gridworld. Across all three settings, we find that achieving high performance on the long tail is difficult. We investigate a variety of solutions, but find only modest, inconsistent benefits to performance on the rarest items. We therefore propose these benchmarks to the community as a challenging evaluation of learning from rare experiences. We hope that our benchmark environments will encourage the development of new RL methods and memory systems, and ultimately the development of agents that can learn from a lifetime of non-uniform experience.

References

Réka Albert and A L Barabasi. Statistical mechanics of complex networks. *ArXiv*, cond-mat/0106096, 2001.

Alex Arenas, Leon Danon, Albert Diaz-Guilera, Pablo M Gleiser, and Roger Guimera. Community analysis in social networks. *The European Physical Journal B*, 38(2):373–380, 2004.

Yuki M. Asano, C. Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv*, abs/1911.05371, 2020.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *ICML*, 2017.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016. URL http://arxiv.org/abs/1612.03801.

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.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 2016.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *ICML*, 2017.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z. Leibo, Jack W. Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *ArXiv*, abs/1606.04460, 2016.

Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, 2002.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv Preprint arXiv:1606.01540*, 2016.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *ArXiv*, abs/1810.12894, 2019.

Joan Bybee and Sandra Thompson. Three frequency effects in syntax. In *Annual Meeting Of The Berkeley Linguistics Society*, volume 23, pp. 378–388, 1997.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv*, abs/2006.09882, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv*, abs/2104.14294, 2021.

Stephanie CY Chan, Samuel Fishman, Anoop Korattikara, John Canny, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *International Conference on Learning Representations*, 2020.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002. ISSN 1076-9757. doi: 10.1613/jair.953. URL http://arxiv.org/abs/1106.1813. arXiv: 1106.1813.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, abs/2002.05709, 2020.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv Preprint arXiv:1810.08272*, 2018.

Elizabeth M Clerkin, Elizabeth Hart, James M Rehg, Chen Yu, and Linda B Smith. Real-world visual statistics and infants' first-learned object names. *Philosophical Transactions Of The Royal Society B: Biological Sciences*, 372 (1711):20160055, 2017.

Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. *arXiv*, abs/1806.06923, 2018a.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI*, 2018b.

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2070–2079, 2017.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv*, abs/1802.01561, 2018.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and Cognitive Architecture: A Critical Analysis. pp. 56, 1988.

Jordan Frank, Shie Mannor, and Doina Precup. Reinforcement learning in the presence of rare events. In *ICML '08*, 2008.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

Florin C. Ghesu, Bogdan Georgescu, Awais Mansoor, Youngjin Yoo, Dominik Neumann, Pragneshkumar B. Patel, R. S. Vishwanath, James M. Balter, Yue Cao, Sasa Grbic, and Dorin Comaniciu. Self-supervised learning from 100 million medical images. *arXiv*, abs/2201.01283, 2022.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv*, abs/2006.07733, 2020.

Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo Ávila Pires, Tobias Pohlen, and Rémi Munos. Neural predictive belief representations. *arXiv*, abs/1811.06407, 2018.

Zhaohan Daniel Guo, Bernardo Ávila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. *arXiv*, abs/2004.14646, 2020.

Jeff Z. HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *arXiv*, abs/2106.04156, 2021.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735, 2020.

Olivier J. Hénaff, A. Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and AäRon Van Den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv*, abs/1905.09272, 2020.

Matteo Hessel, Joseph Modayil, H. V. Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech M. Czarnecki, Simon Schmitt, and H. V. Hasselt. Multi-task deep reinforcement learning with popart. In *AAAI*, 2019.

Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent. *arXiv preprint arXiv:1910.00571*, 2019.

Felix Hill, Olivier Tieleman, Tamara Von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark. Grounded language learning fast and slow. *arXiv Preprint arXiv:2009.01719*, 2020.

Geoffrey E. Hinton. To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–47, 2007.

Sepp Hochreiter and JÜRgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv Preprint arXiv:1611.05397*, 2016.

Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6 (1):27, March 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0192-5. URL `https://doi.org/10.1186/s40537-019-0192-5`.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018.

Michael Kearns, Y. Mansour, and A. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. 1999.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktaschel. A survey of generalisation in deep reinforcement learning. *arXiv*, abs/2111.09794, 2021.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv:1711.00350 [cs]*, June 2018. URL http://arxiv.org/abs/1711.00350. arXiv: 1711.00350.

Andrew Lampinen, Stephanie Chan, Andrea Banino, and Felix Hill. Towards mental time travel: A hierarchical memory for reinforcement learning agents. *Advances in Neural information Processing Systems*, 34, 2021.

Tor Lattimore and Marcus Hutter. Near-optimal pac bounds for discounted mdps. *Theor. Comput. Sci.*, 558:125–143, 2014.

Máté Lengyel and Peter Dayan. Hippocampal contributions to control: The third way. In *NIPS*, 2007.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Longxin Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. *arXiv*, abs/2110.05025, 2021.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings Of The IEEE international Conference on Computer Vision*, pp. 3730–3738, 2015.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2532–2541, 2019.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419, 1995.

Jovana Mitrovic, Brian McWilliams, Jacob Walker, Lars Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms. *arXiv*, abs/2010.07922, 2021.

Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11174–11182, 2020.

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

Ashvin V. Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in Neural information Processing Systems*, 31, 2018.

Mark E. J. Newman. Power laws, Pareto distributions and Zipf's law. 2004.

Luke Oakden-Rayner, Jared Dunnmon, Gustavo Carneiro, and Christopher Ré. Hidden Stratification Causes Clinically Meaningful Failures in Machine Learning for Medical Imaging. *arXiv:1909.12475 [cs, stat]*, November 2019. URL http://arxiv.org/abs/1909.12475. arXiv: 1909.12475.

Aäron Van Den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv*, abs/1807.03748, 2018.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *ArXiv*, abs/1703.01310, 2017.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Çaglar Gülçehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Manfred Otto Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. *arXiv*, abs/1910.06764, 2020.

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544, 2016.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2018.

Steven T. Piantadosi. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21(5):1112–1130, October 2014. ISSN 1069-9384. doi: 10.3758/s13423-014-0585-6. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4176592/.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International Conference on Machine Learning*, pp. 2827–2836. PMLR, 2017.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Samuel Ritter, Ryan Faulkner, Laurent Sartran, Adam Santoro, Matthew Botvinick, and David Raposo. Rapid task-solving in novel environments. In *International Conference on Learning Representations*, 2020.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization. *arXiv:1911.08731 [cs, stat]*, April 2020. URL http://arxiv.org/abs/1911.08731. arXiv: 1911.08731.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2016.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021.

Linda B Smith and Lauren K Slone. A developmental approach to machine learning? *Frontiers in Psychology*, 8: 2124, 2017.

Linda B Smith, Swapnaa Jayaraman, Elizabeth Clerkin, and Chen Yu. The developing infant creates a curriculum for statistical learning. *Trends in Cognitive Sciences*, 22(4):325–336, 2018.

Nimit S. Sohoni, Jared A. Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No Subclass Left Behind: Fine-Grained Robustness in Coarse-Grained Classification Problems. *arXiv:2011.12945 [cs]*, November 2020. URL http://arxiv.org/abs/2011.12945. arXiv: 2011.12945.

H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Manfred Otto Heess, Dan Belov, Martin A. Riedmiller, and Matthew M. Botvinick. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv*, abs/1909.12238, 2020.

Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pp. 935–942, New York, NY, USA, June 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273614. URL https://doi.org/10.1145/1273496.1273614.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, abs/1706.03762, 2017.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, and Others. Starcraft II: A new challenge for reinforcement learning. *arXiv Preprint arXiv:1708.04782*, 2017.

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, and Others. Unsupervised predictive memory in a goal-directed agent. *arXiv Preprint arXiv:1803.10760*, 2018.

Yifan Wu, G. Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. *arXiv*, abs/1810.04586, 2019.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, P. Abbeel, and Yang Gao. Mastering Atari games with limited data. *arXiv*, abs/2111.00210, 2021.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv*, abs/2006.10742, 2021a.

Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv*, abs/1804.06893, 2018.

Yifan Zhang, Bryan Hooi, Lanqing Hong, and Jiashi Feng. Test-agnostic long-tailed recognition by test-time aggregating diverse experts with self-supervision. *arXiv*, abs/2107.09249, 2021b.

Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *Proceedings Of The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 915–922, 2014.

George Kingsley Zipf. *The psycho-biology of language: An introduction to dynamic philology*. routledge, 1936.

A Appendix

A.1 Zipfian distributions

The following python code snippet generates Zipfian distributions for a specified exponent:

```
import numpy

def zipfian_dist(num_entities, exponent=1.):
  vals = 1./(numpy.arange(1, num_entities + 1))**exponent
  return vals / numpy.sum(vals)
```

A.2 Using the benchmarks

A.2.1 General principles for training and evaluation

In each of our environments, it is necessary to create separate copies of the environment for each training level and testing level in a given split (see below).

To evaluate, you should run at least three seeds (independent runs) per condition and evaluate after training has converged to relatively stable performance. You should report average performance of the median seed(s) over a relatively large set of episodes (as a rule of thumb, at least 1000 episodes per evaluation level).

While we encourage exploring variations on our benchmarks (for example increasing or decreasing the Zipf exponent), to avoid confusion we ask that you state any modifications very clearly in your publications or reports, and do not use unqualified versions of our split names when describing your results.

A.3 Installing and running the benchmarks

Our benchmarks are released online at redacted

Once you have cloned the repository, you can install most the needed dependencies for all three benchmarks by running the following commands:

```
python3 -m venv zipf
source zipf/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
```

However, note that this will install dependencies for all three benchmarks, and will not install some dependencies which are not listed in pip, see below.

A.3.1 ZIPF'S PLAYROOM

To use Zipf's playroom, you will also need to download and install Docker. Once you have done so, you can create an environment as follows:

```python
import zipfs_playroom
env_settings = zipfs_playroom.EnvironmentSettings(
  seed=1, level_name='lift/lift_shape_zipf3')
env = zipfs_playroom.load_from_docker(
  name=FLAGS.docker_image_name, settings=env_settings)
obs = env.reset()
```

Note that you will have to instantiate different environments for training and test level (and train on several distinct levels for the 'put' tasks), see Table 5.

Task split	Train levels	Test levels
Lifting task	'lift/lift_shape_zipf3'	'lift/lift_shape_uniform' 'lift/lift_shape_uniform_rare'
Putting task	'put/lift_shape_zipf2' 'put/put_near_bed_tray_frequent' 'put/put_on_bed_tray_frequent'	'put/put_on_bed_tray_all' 'put/put_on_bed_tray_rare'

Table 5: Train and test level splits in Zipf's Playroom.

A.3.2 ZIPF'S LABYRINTH

To use Zipf's Labyrinth, you will also need to install DM-Lab. Once you have done so, you can instantiate an environment as follows:

```python
import zipfs_labyrinth
env = zipfs_labyrinth(level_name='forward_zipf')
obs = env.reset()
```

Note that you will have to instantiate different environments for each training and testing level, see Table 6.

Task split	Train levels	Test levels
Forward Zipf	'forward_zipf'	'uniform' 'forward_rare'
Reversed Zipf	'reversed_zipf'	'uniform' 'reversed_rare'

Table 6: Train and test level splits in Zipf's Labyrinth.

A.3.3 ZIPF'S GRIDWORLD

To use Zipf's Gridworld, you can create an

```python
import zipfs_gridworld
env = zipfs_gridworld.simple_builder(level_name='zipf_2')
obs = env.reset()
obs2 = env.step(6)
```

Note that you will have to instantiate different environments for each training and testing level, see Table 7.

Task split	Train levels	Test levels
Zipf 2	'zipf_2'	'uniform' 'rare'

Table 7: Train and test level splits in Zipf's Gridworld.

A.4 Environment details

A.4.1 Zipf's Playroom

Zipf's Playroom is implemented using Unity. The agent receives a visual observation of $96 \times 72 \times 3$ pixel RGB images, and a language observation that is tokenized at the word-level. The agent exerts relatively low-level control, using a discrete action space of 46 actions that allow several different magnitudes of movement forward and back, strafing side-to-side, rotating left and right and looking up and down, and grabbing and manipulating held objects. The physics is relatively realistic, for example the agent has momentum in its movements. The agent acts at 7.5 actions per environment second (30 environment FPS, with 4 action repeats between agent steps). Zipf's playroom runs at around 900 FPS (clock time), comparable to DMLab when running on comparable hardware (an NVIDIA Quadro K600 GPU).

A.4.2 Zipf's Labyrinth

This environment was used as in Beattie et al. (2016), except that the distribution of levels was Zipfian. For further details, see the original publication. The environments run at about 700-1000 FPS (depending on the level) on a Linux desktop with a 6-core Intel Xeon 3.50GHz CPU and an NVIDIA Quadro K600 GPU (per the original publication). The Zipf distributions were applied to the levels in the original order:

```
LEVELS_DMLAB30_FORWARD = [
    'rooms_collect_good_objects_train', 'rooms_exploit_deferred_effects_train'
       ,
    'rooms_select_nonmatching_object', 'rooms_watermaze',
    'rooms_keys_doors_puzzle', 'language_select_described_object',
    'language_select_located_object', 'language_execute_random_task',
    'language_answer_quantitative_question', 'lasertag_one_opponent_small',
    'lasertag_three_opponents_small', 'lasertag_one_opponent_large',
    'lasertag_three_opponents_large', 'natlab_fixed_large_map',
    'natlab_varying_map_regrowth', 'natlab_varying_map_randomized',
    'skymaze_irreversible_path_hard_v2', 'skymaze_irreversible_path_varied_v2'
       ,
    'psychlab_arbitrary_visuomotor_mapping', 'psychlab_continuous_recognition'
       ,
    'psychlab_sequential_comparison', 'psychlab_visual_search',
    'explore_object_locations_small', 'explore_object_locations_large',
    'explore_obstructed_goals_small', 'explore_obstructed_goals_large',
    'explore_goal_locations_small', 'explore_goal_locations_large',
    'explore_object_rewards_few', 'explore_object_rewards_many'
]
# Reverse distribution
LEVELS_REVERSED = list(reversed(LEVELS_DMLAB30_FORWARD))
```

Note that this ordering clusters the tasks by type, which adds a degree of hierarchy to the skewed experience.

A.4.3 Zipf's Gridworld

The agent observes 7×7 grid squares around it, rendered at a 9×9 pixel resolution pre square, for a total visual observation size of 63×63 pixels. The top left square of this observation is replaced with a heads-up display showing the target object, displayed on a distinctive light-gray background. The agent can move to any of the 8 adjacent squares, including diagonals. If it move onto an object, the episode immediately ends, and the agent is rewarded if the object it moved to is the target object. Otherwise, it is not rewarded. The episode also terminates with 0 reward if the agent has not touched an object within 100 steps (although this is quite unlikely even under a random policy, as the object density is high).

The objects for each map are created by combining the following 15 colors and 15 shapes, subject to the constraint that all objects are distinct along at least one dimension within each map (but no constraints across maps):

```
COLORS = [
    "red", "green", "blue", "purple", "orange",
    "yellow", "brown", "pink", "cyan", "dark_green",
    "dark_red", "dark_blue", "teal", "lavender", "rose"
]
SHAPES = [
    "triangle", "empty_square", "plus", "inverse_plus", "ex",
    "inverse_ex", "circle", "empty_circle", "tee", "upside_down_tee",
    "h", "u", "upside_down_u", "vertical_stripes", "horizontal_stripes"
]
```

Objects are placed such that the agent can navigate between any two points in the map without touching an object, to ensure that all rooms and objects are reachable.

Zipf's Gridworld is implemented using the pycolab gridworld engine https://github.com/deepmind/pycolab. In a simple benchmark (stepping the environment with a fixed action), we measured that Zipf's Gridworld runs at around 4,700 FPS in a single thread on an Intel Xeon W-2135 CPU.

A.5 Experiment hyperparameters

All agents	
language encoder type	LSTM
language encoder embedding size	32
language encoder hidden size	32
vision encoder type	ResNet
vision encoder output channels	(16, 32, 32)
vision encoder resnet blocks	(2, 2, 2)
encoder mixing operation	flatten+concat
LSTM memory core	
hidden size	512
Transformer memory core	
d_model	512
num layers	4
num attention heads	8
Self-supervised learning	
language reconstruction network	LSTM
visual reconstruction network	deconv network
lang recon hyper params	as encoder
visual recon hyper params	as encoder
lang recon loss	cross-entropy
visual recon loss	sigmoid cross-entropy
BYOL forward hidden	100
BYOL projection hidden	200
BYOL prediction hidden	100
BYOL projection size	50

Table 8: Architecture hyperparameters for all agents.

A.5.1 IMPALA AGENT

	Zipf's Labyrinth	Zipf's Labyrinth	Zipf's Gridworld
Image Width	96	96	96
Image Height	72	72	72
Action Repeats	4	4	4
Unroll Length	128	128	128
Discount (γ)	0.99	0.99	0.99
Baseline loss scaling	0.6	0.5	0.59
Entropy cost	$1e-4$	0.01	$9.4e-5$
Action Repeats	4	4	4
Optimizer	Adam	Adam	Adam
Learning rate	$3e-4$	$1e-4$	$3e-4$

A.5.2 V-MPO AGENT

We note that V-MPO hyperparameters may not be optimally tuned for these settings, as it was unable to achieve high performance on the putting tasks, even in training (thus we have omitted those results). While V-MPO achieved relatively high training performance on the lifting tasks, it was still noticeably worse than the other algorithms. However, we were unable to find better hyperparameters for it within the sweeps we considered. Nevertheless, the comparison between V-MPO and the other algorithms should be interpreted cautiously.

Parameter	Value
Agent discount	0.995
Image width	96
Image height	72
Number of action repeats	4
Number of LSTM layers	2
MPO learning rate	$3 * 10^{-3}$
Net learning rate	$3 * 10^{-5}$
T_{target}	0.5
ϵ_η	0.1
ϵ_α	0.1

Table 5: Settings for V-MPO

A.5.3 R2D2 AGENT

A.5.4 ZIPF'S PLAYROOM

In *Zipf's Playroom*, there is a distinction between hyperparameters on the *lifting* tasks and the *putting* tasks.

Tasks type	Lifting tasks	Putting tasks
Number of actors	256	
Sequence length	128(prefix of $l = 20$ burn-in)	
Replay buffer size	10^5	
Minibatch size	32	
Importance sampling exponent	0.6	
Priority weight (max)	0.5	0.9
Exploration ε	0.125	0.15
Discount γ	0.99	
Lambda-returns λ	0.8	
Optimizer	AdamW (Loschilov & Hutter, 2017)	
Optimizer learning rate	$5 * 10^{-4}$	$5 * 10^{-5}$
Optimizer ε	$1.25 * 10^{-6}$	
Weight decay	10^{-4}	
Max optimizer gradient norm	0.5	
Target network update interval	400 updates	
Value function rescaling	$h(x) = \text{sign}(x)(\sqrt{\lvert x\rvert + 1} - 1) + \epsilon x, \epsilon = 10^{-3}$	

A.5.5 ZIPF'S LABYRINTH

Number of actors	256
Sequence length	128(prefix of $l = 20$ burn-in)
Replay buffer size	10^5
Minibatch size	32
Importance sampling exponent	0.6
Priority weight (max)	0.9
Exploration ε	10^{-3}
Discount γ	0.997
Lambda-returns λ	0.8
Optimizer	AdamW (Loschilov & Hutter, 2017)
Optimizer learning rate	$3 * 10^{-4}$
Optimizer ε	$1.25 * 10^{-6}$
Weight decay	10^{-4}
Max optimizer gradient norm	0.5
Target network update interval	400 updates
Value function rescaling	$h(x) = \text{sign}(x)(\sqrt{\lvert x\rvert + 1} - 1) + \epsilon x, \epsilon = 10^{-3}$

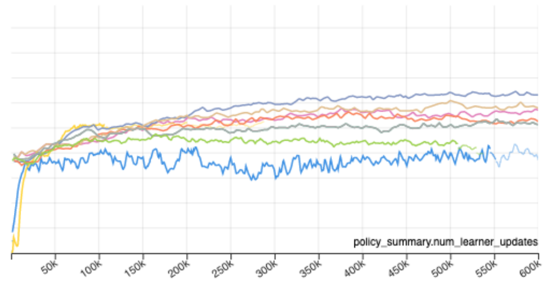
A.5.6 ZIPF'S GRIDWORLD

Number of actors	256
Sequence length	128(prefix of $l = 3$ burn-in)
Replay buffer size	10^5
Minibatch size	32
Importance sampling exponent	0.6
Priority weight (max)	0.9
Exploration ε	0.1
Discount γ	0.9
Lambda-returns λ	0.3
Optimizer	AdamW (Loschilov & Hutter, 2017)
Optimizer learning rate	$3 * 10^{-4}$
Optimizer ε	$1.25 * 10^{-6}$
Weight decay	10^{-4}
Max optimizer gradient norm	0.5
Target network update interval	10 updates
Value function rescaling	$h(x) = \text{sign}(x)(\sqrt{\lvert x\rvert + 1} - 1) + \epsilon x, \epsilon = 10^{-3}$

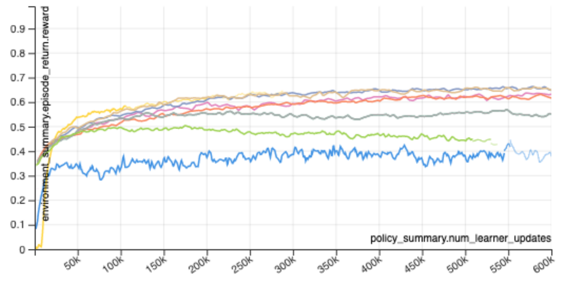
B SUPPLEMENTAL ANALYSES

B.1 TRAIN PERFORMANCE

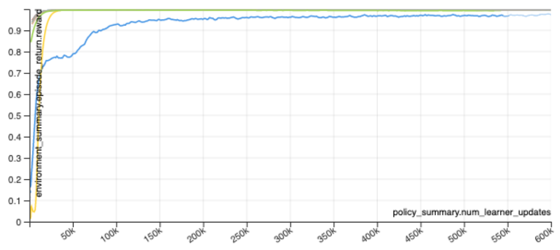
Train performance, i.e. assessed on the Zipfian distributions that the agents encountered during training, was generally very high and obscured the relatively poor performance on rare scenarios. Train performance for Zipf's Labyrinth is reported directly in Table 2 under 'All tasks (Zipfian)'.



(a) Rare - uniform.



(b) All - uniform.



(c) All - Zipfian (training distribution).

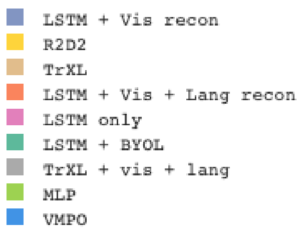


Figure 6: Learning curves for lifting task in Zipf's Playroom.

		Lifting task	Putting task	
		All items (Zipf $\alpha = 3$)	Lifting (Zipf $\alpha = 2$)	Putting (frequent only)
RL	IMPALA	0.999	0.993	0.987
algorithm	V-MPO	0.970	0.791	0.091
SSL	Vis recon	0.999	0.992	0.985
(IMPALA +..)	Vis+lang recon	0.999	0.990	0.976
	BYOL	0.999	0.989	0.965
Memory	LSTM	0.999	0.993	0.987
system	Transformer	0.999	0.991	0.963
(IMPALA +..)	Transformer + recon	0.999	0.973	0.801
	Prioritized (R2D2)	1.000	0.995	0.977

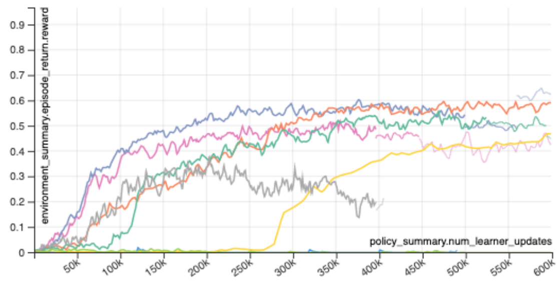
Table 9: Training performance in Zipf's Playroom.

	All maps and objects (Zipf $\alpha = 2$)
IMPALA	0.851
IMPALA + Vis recon	0.989
IMPALA + BYOL	0.751
R2D2	0.672

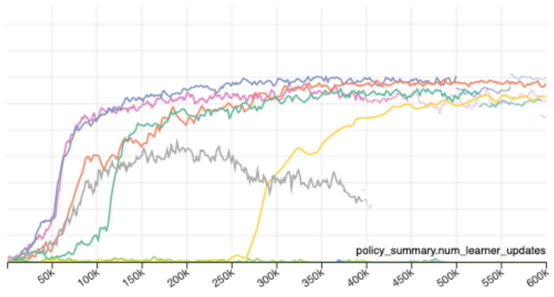
Table 10: Training performance in Zipf's Gridworld. Median performance (across time) of median run for each condition.

C Learning curves

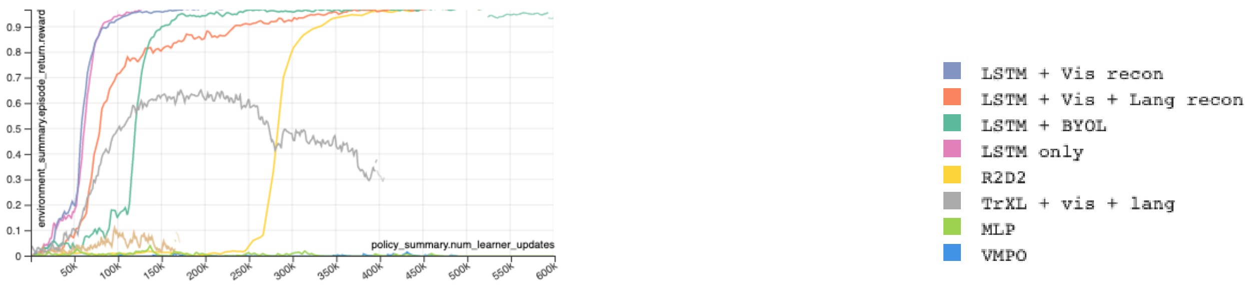
Learning curves are displayed for Zipf's Playroom (Figs 6-7), Zipf's Labyrinth (Fig 9), and Zipf's Gridworld (Fig 10).



(a) Rare - uniform.

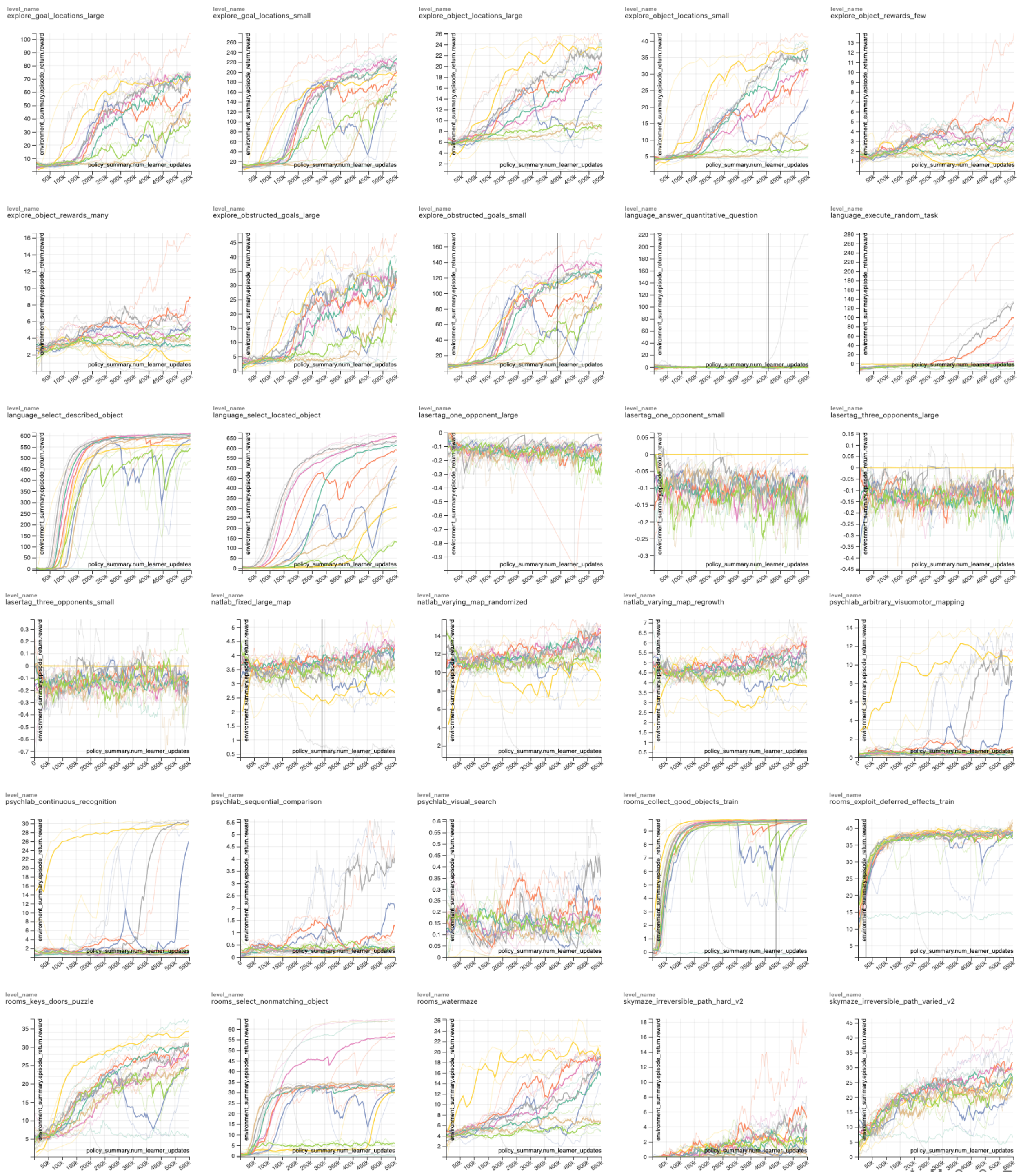


(b) All - uniform.



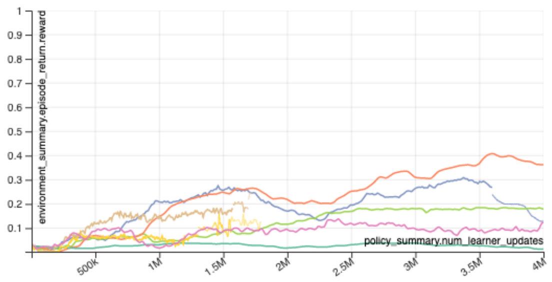
(c) All - Zipfian (training distribution).

Figure 7: Learning curves for putting task in Zipf's Playroom.

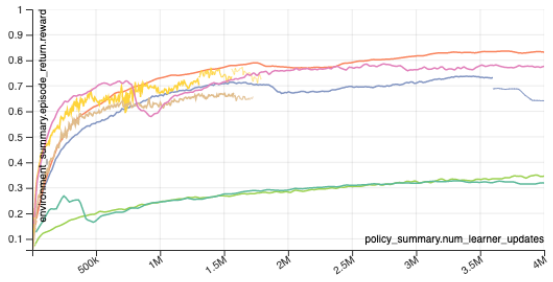


🟩 LSTM + BYOL
🟥 LSTM + Vis + Lang recon
🟦 LSTM + Vis recon
🟪 LSTM only
🟩 MLP
🟨 R2D2
🟫 TrXL
⬜ TrXL + vis + lang

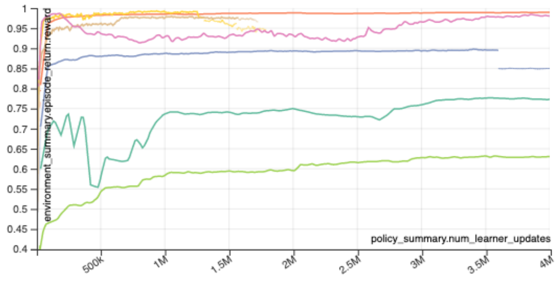
Figure 9: Learning curves for Zipf's Labyrinth (training on forward Zipf distribution).



(a) Rare - uniform.



(b) All - uniform.



(c) All - Zipfian (training distribution).

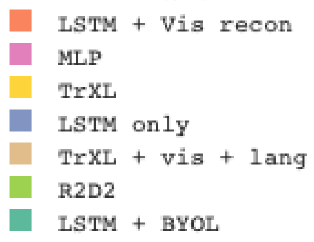


Figure 10: Learning curves for Zipf's Gridworld.