Human-Timescale Adaptation in an Open-Ended Task Space

Jakob Bauer [†1] **Kate Baumli** [†1] **Feryal Behbahani** [†1] **Avishkar Bhoopchand** [†1] **Nathalie Bradley-Schmieg** [†1] **Michael Chang** [†1] **Natalie Clay** [†1] **Adrian Collister** [†1] **Vibhavari Dasagi** [†1] **Lucy Gonzalez** [†1] **Karol Gregor** [†1] **Edward Hughes** [†1] **Sheleem Kashem** [†1] **Maria Loks-Thompson** [†1] **Hannah Openshaw** [†1] **Jack Parker-Holder** [†1] **Shreya Pathak** [†1] **Nicolas Perez-Nieves** [†1] **Nemanja Rakicevic** [†1] **Tim Rocktäschel** [†1] **Yannick Schroecker** [†1] **Satinder Singh** [†1] **Jakub Sygnowski** [†1] **Karl Tuyls** [†1] **Sarah York** [†1] **Alexander Zacherl** [†1] **Lei Zhang** [†1]

Abstract

Foundation models have shown impressive adaptation and scalability in supervised and self-supervised learning problems, but so far these successes have not fully translated to reinforcement learning (RL). In this work, we demonstrate that training an RL agent at scale leads to a general in-context learning algorithm that can adapt to open-ended novel embodied 3D problems as quickly as humans. In a vast space of held-out environment dynamics, our adaptive agent (AdA) displays on-the-fly hypothesis-driven exploration, efficient exploitation of acquired knowledge, and can successfully be prompted with first-person demonstrations. Adaptation emerges from three ingredients: (1) meta-reinforcement learning across a vast, smooth and diverse task distribution, (2) a policy parameterised as a large-scale attention-based memory architecture, and (3) an effective automated curriculum that prioritises tasks at the frontier of an agent's capabilities. We demonstrate characteristic scaling laws with respect to network size, memory length, and richness of the training task distribution. We believe our results lay the foundation for increasingly general and adaptive RL agents that perform well across ever-larger open-ended domains.

1. Introduction

Meta-RL has been shown to be effective for fast in-context adaptation (e.g. Yu et al. (2020); Zintgraf (2022)). However, meta-RL has had limited success in settings where the reward is sparse and the task space is vast and diverse (Yang et al., 2019). Outside RL, *foundation models* in semi-supervised learning have generated significant interest (Bommasani et al., 2021) due to their ability to adapt in few shots from demonstrations across a broad range of tasks. These models are designed to provide a strong foundation of general knowledge and skills that can be built upon and adapted to new situations via fine-tuning or prompting with demonstrations (Brown et al., 2020). Crucial to this success has been attention-based memory architectures like Transformers (Vaswani et al., 2017), which show power-law scaling in performance with the number of parameters (Tay et al., 2022).

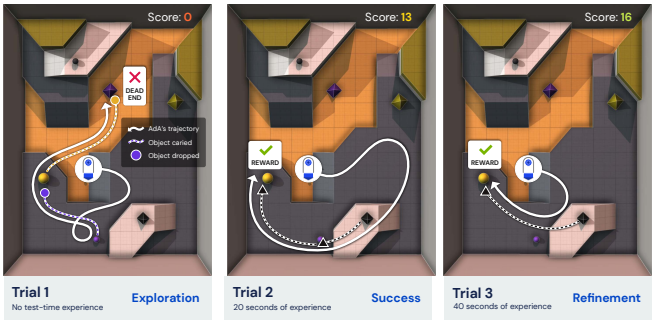


Figure 1: **Adaptation in minutes.** Example trajectories of our agent (AdA) solving a held-out task in a complex 3D environment within minutes of test-time experience without any further agent training. Initial trials (Exploration) show a policy that uncovers hidden environment dynamics. After just seconds of test-time experience (Success), AdA finds a valid solution to the task. Later (Refinement), it improves this solution, gradually finding a more rewarding behaviour. The solid white lines show agent movement. The dashed coloured lines show the agent carrying an object of the corresponding colour. For a full description of the task, see Figure E.1. Videos of AdA's behaviour are available on our microsite and accompanying results reel.

In this work, we pave the way for training an RL foundation model; that is, an agent that has been pre-trained on a vast task distribution and that, at test time, can adapt few-shot to a broad range of downstream tasks. We introduce *Adaptive Agent* (AdA), an agent capable of human-timescale adaptation in a vast open-ended task space with sparse rewards. AdA does not require any prompts (Reed et al.,

[†]Alphabetical order, see Adaptive Agent Team for contributions [1]DeepMind. Correspondence to: Feryal Behbahani <feryal@google.com>, Edward Hughes <edwardhughes@google.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

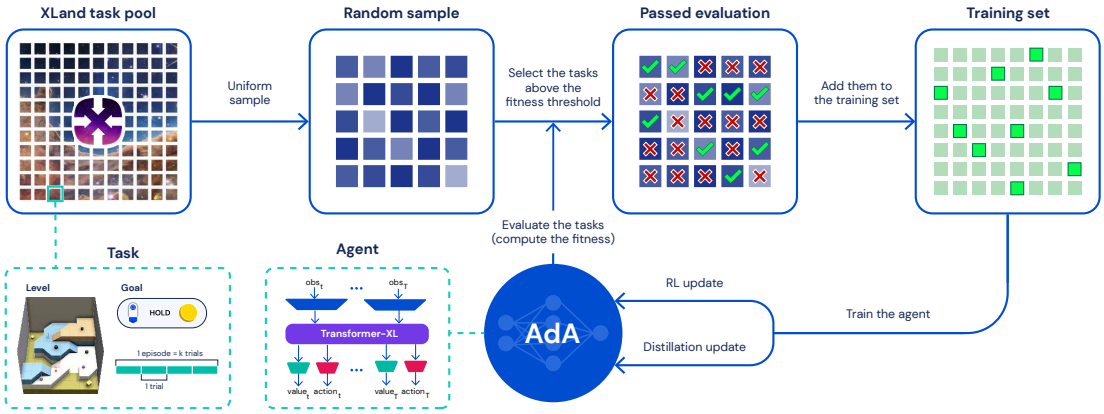


Figure 2: **Training our Adaptive Agent (AdA).** We train a large Transformer model with meta-RL in XLand. During training, tasks are uniformly sampled, and subsequently filtered to produce an ever-changing training pool of tasks at the frontier of the agent's capabilities. After training on these tasks, the agent is capable of adapting to unseen hand-authored tasks as effectively and efficiently as humans.

2022), fine-tuning (Lee et al., 2022) or access to offline datasets (Laskin et al., 2022; Reed et al., 2022). Instead, AdA exhibits hypothesis-driven exploratory behaviour, using information gained on-the-fly to refine its policy and to achieve close to optimal performance. AdA acquires knowledge efficiently, adapting in minutes on challenging held-out sparse-reward tasks in a partially-observable 3D environment with a first-person pixel observation. A human study confirms that the timescale of AdA's adaptation is comparable to that of trained human players. AdA's adaptation behaviour in a representative held-out task can be seen in Figure 1. AdA can also achieve improved performance through zero-shot prompting with first-person demonstrations, analogously to large language models.

We use Transformers as an architectural choice to scale in-context fast adaptation via model-based RL^2 (Duan et al., 2017; Wang et al., 2016; Melo, 2022). Foundation models typically require large, diverse datasets to achieve their generality (Sun et al., 2017; Mahajan et al., 2018; Brown et al., 2020; Zhai et al., 2022; Schuhmann et al., 2022). To make this possible in an RL setting, where agents collect their own data, we extend the recent XLand environment (OEL Team et al., 2021), producing a vast open-ended world with over 10^{40} possible tasks. These tasks require a range of different online adaptation capabilities, including experimentation, navigation, coordination, division of labour and coping with irreversibility. Given the wide range of possible tasks, we make use of adaptive auto-curricula, which prioritise tasks at the frontier of an agent's capabilities (OEL Team et al., 2021; Jiang et al., 2021a). Finally, we make use of distillation (Schmitt et al., 2018), which enables scaling to models with over 500M parameters, to the best of our knowledge the largest model trained from scratch with RL at the time of publication (Ota et al., 2021). See Figure 2 for an overview of our method.

2. Adaptive Agent (AdA)

To achieve human timescale adaptation across a vast and diverse task space, we propose a general and scalable approach for memory-based meta-RL, producing an *Adaptive Agent* (AdA). We train and test AdA in XLand 2.0, an environment supporting procedural generation of diverse 3D worlds and multi-player games, with rich dynamics that necessitate adaptation. Our training method combines three key components: a curriculum to guide the agent's learning, a model-based RL algorithm to train agents with large-scale attention-based memory, and distillation to enable scaling. For a detailed discussion of related work see Appendix 4. An overview of our approach is shown in Figure 2. In the following sections, we describe each component and its contribution to efficient few-shot adaptation.

2.1. Open-ended task space: XLand 2.0

In order to demonstrate fast adaptation across an open-ended task space, we extend the procedurally-generated 3D environment XLand (OEL Team et al., 2021). In XLand, a task consists of a game, a world, and a list of co-player policies (if any). The game specifies a goal per player, defined as a boolean function (predicate) on the environment state. A player receives reward if and only if the goal is satisfied. The world specifies a static floor topology, objects the players can interact with, and spawn locations for the players. Each player observes the world, and any co-players therein, via a first-person pixel observation.

XLand 2.0 extends XLand with a system called *production rules*. Each production rule expresses an additional environment dynamic, leading to a much richer and more diverse array of transition functions than in XLand. Production rules support a wide variety of different dynamics, including tasks inspired by chemistry experiments, the video game Overcooked, the 2048 browser game, soccer prac-

tice, antimatter, factory machines, tool use and the proverbial needle in the haystack. See Tables I.1 and I.2 for a full description of 58 representative tasks in XLand 2.0. The production rules system can be thought of as a domain-specific language to express this diverse array of dynamics. Each production rule consists of a `condition`, which is a predicate, for example `near(yellow sphere,black cube)`, and a (possibly empty) list of `spawns`, which are objects, like `purple cube, black cube`.

When `condition` is satisfied, the objects present in `condition` get removed from the environment, and the ones in `spawns` appear. Each task can have multiple production rules, and each rule can be observable to players, or partially or fully masked, depending on the task configuration, as described in Appendix D.1. For a more detailed understanding of how human-interpretable tasks can be represented via production rules, see Figures E.1–E.4. We visualise the XLand 2.0 task space in Figure D.1, indicating the different cognitive challenges that clusters of tasks pose for a player.

When training AdA, instead of procedurally generating tasks on the fly, we pre-sample a large pool of tasks, for efficiency and to reduce variance (see Appendix D.2).

2.2. Auto-curriculum learning

Given the vastness and diversity of our pre-sampled task pool, it is challenging for an agent to learn effectively with uniform sampling. Most randomly sampled tasks are likely to be too hard (or too easy) to benefit an agent's learning. Instead, we use automatic approaches to select "interesting" tasks at the frontier of the agent's capabilities, analogous to the "zone of proximal development" in human cognitive development (Vygotsky, 1978). We propose two alternative approaches, which lead to an emergent curriculum, selecting tasks with increasing complexity over time.

No-op filtering. We extend the dynamic task generation method proposed in OEL Team et al. (2021). When a new task is sampled from the pool, it is first evaluated to as follows. We run AdA's policy and a "No-op" control policy (which takes no action in the environment) on the task for a number of episodes. The task is then used for training (for 30 trials) if and only if the scores of the two policies meet a number of conditions. We expanded the list of conditions from the original no-op filtering and used normalised thresholds to account for different trial durations. See Appendix G.6 for further details.

Prioritised level replay (PLR). We modify robust PLR (Jiang et al., 2021a) to fit our setup. By contrast to no-op filtering, PLR uses a *fitness score* (Schmidhuber, 1991) that approximates the agent's regret for a given task. We consider several potential estimates for agent regret, ranging from TD-errors (Jiang et al., 2021b), to novel approaches using dynamics-model errors from AdA. PLR operates by maintaining a fixed-sized archive containing tasks with the highest fitness. It can be seen as a form of filtering, using a dynamic criterion (the lowest fitness value of the archive). To apply PLR in our heterogeneous task space, we normalise fitness at each trial index by using rolling means and variances, and use the mean per-timestep fitness value rather than the sum, to account for varying trial duration. Since we are interested in tasks at the frontier of an agent's capabilities after adaptation, we use only the fitness from the last trial. See Appendix G.6 for further details.

2.3. Meta-RL

We use a black-box meta-RL problem setting (Duan et al., 2017; Wang et al., 2016). We define the task space to be a set of partially-observable Markov decision processes (POMDPs). In black-box meta-reinforcement learning, an episode of experience for an agent consists of multiple consecutive interactions with the same task (called trials), with the idea that the agent can learn to self-improve its policy on later trials based on the memory of information gleaned in earlier trials. For a given task, we define a *trial* to be any sequence of transitions from an initial state s_0 to a terminal state s_T. Tasks terminate if and only if a certain time period $T \in [10\text{s}, 40\text{s}]$ has elapsed, specified per-task. The environment ticks at 30 frames-per-second and the agent observes every 4^{th} frame. An *episode* consists of a sequence of k trials for a given task. At trial boundaries, the task is reset to an initial state. In our domain, initial states are deterministic except for the rotation of the agent, which is sampled uniformly at random. The memory of the agent is not reset at trial boundaries, but is reset at episode boundaries. For full details on AdA's meta-RL method, see Appendix G.1. We train AdA using the Muesli algorithm (Hessel et al., 2021) with minor modifications to fit our meta-RL setting, as described in Appendix F.1.

2.4. Memory architecture

Memory is a crucial component for adaptation as it allows the agent to store and recall information learned and experienced in the past. In order for agents to effectively adjust to the changes in task requirements, memory should allow the agent to recall information from both the very recent and the more distant past. While slow gradient-based updates are able to capture the latter, they are often not fast enough to capture the former, i.e. fast adaptation. The majority of work on memory-based meta-RL has relied on RNNs as a mechanism for fast adaptation (Parisotto, 2021). In this work, we show that RNNs are not capable of adaptation in our challenging partially-observable embodied 3D task space. We experiment with two memory architectures to address this problem, as follows.

Firstly, *RNN with Attention* stores a number of past activations (in our case 64) in an episodic memory and attends over it, using the current hidden state as a query. The output of the attention module is then concatenated with the hidden state and fed into the RNN. We increase effective memory length of the agent by storing only every 8^{th} activation in its episodic memory. Secondly, we consider *Transformer-XL (TXL)* (Dai et al., 2019), a variant of the Transformer architecture (Vaswani et al., 2017) which enables the use of longer, variable-length context windows to increase the model's ability to capture long-term dependencies. To stabilise the training of Transformers with RL, we follow Parisotto et al. (2020) in performing normalisation *before* each layer, and use gating on the feedforward layers as in Shazeer (2020). Both memory modules operate on a sequence of learned timestep embeddings, and produce a sequence of output embeddings that are fed into the Muesli architecture, as shown in Figure F.1 with a Transformer-XL module. Transformer-XL is the default memory architecture in all our experiments unless stated otherwise.

To go beyond few shots, we propose a simple modification to our Transformer-XL architecture to increase the effective memory length without additional computational cost. Since observations in visual RL environments tend to be highly temporally correlated, we sub-sample the sequence as described for RNN with Attention, allowing the agent to attend over 4 times as many trials. To ensure that observations which fall between the sub-sampled points can still be attended to, we first encode the entire trajectory using an RNN with the intention of summarising recent history at every step. The additional RNN encoding does not affect the performance of our Transformer-XL variant but enables longer range memory (see Appendix H.6).

2.5. Distillation

For the first four billion steps of training, we use an additional distillation loss (Schmidhuber, 1992; Schmitt et al., 2018; Czarnecki et al., 2019) to guide AdA's learning with the policy of a pre-trained teacher, in a process known as kickstarting; iterating this process leads to a *generational training* regime (OEL Team et al., 2021; Wang et al., 2021). The teacher is pre-trained from scratch via RL, using an identical training procedure and hyperparameters as AdA, apart from the lack of initial distillation and a smaller model size (23M Transformer parameters for the teacher and 265M for multi-agent AdA).

Unlike aforementioned prior work, we do not employ shaping rewards or Population Based Training (Jaderberg et al., 2017) in earlier generations. During distillation, AdA acts according to its own policy and the teacher provides target logits given the trajectories observed by AdA. Distillation allows us to amortise an otherwise costly initial training period, and it allows the agent to overcome harmful representations acquired in the initial phases of training (Cetin et al., 2022); see Section 3.6. For details of how we integrate the distillation loss with Muesli, see Appendix G.2.

3. Experiments and Results

We evaluate our agents in two distinct regimes: on a set of 1000 *test tasks* sampled from the same distribution as the training tasks, and on a set of 30 single-agent and 28 multi-agent *hand-authored probe tasks*. A rejection sampling procedure guarantees that the procedural test tasks and probe tasks are outside the training set. Explicitly, whenever we create a probe or test task, we check whether its combination of goal and production rules is present in the set of pre-sampled training tasks. If it is, the probe or test task is rejected and we construct a new one. The probe tasks represent situations that are particularly intuitive to humans, and deliberately cover a wide range of qualitatively different adaptation behaviours. Example probe tasks are depicted in Figures E.1–E.4 in the Appendix, and a full description of every probe task is in Appendix I.

The total achievable reward on each task varies, so whenever we present aggregated results on the test or hand-authored task set, we normalise the total per-trial reward for each task against the reward obtained by fine-tuning AdA on the respective task set (see Appendix G for details). We refer to this normalised reward as a *score*. We stipulate that an adaptive agent must have two capabilities: zero-shot generalisation and few-shot adaptation. Zero-shot *generalisation* is assessed by the score in the case of only being given 1 trial of interaction with a held-out task. Few-shot *adaptation* is assessed by the improvement in score as the agent is given progressively more trials (k) of interaction with the task. More precisely, for each k we report the score in the last trial, showing whether or not an agent is able to make use of additional experience on-the-fly to perform better, i.e. measuring adaptation. We aggregate scores across a task set using (one or more) percentiles. When presenting individual probe tasks we report unnormalised total last trial rewards per task for agents and for human players where applicable. For more details, see Appendix E.

The space of training configurations for AdA is large, comprising model size, auto-curriculum, memory architecture, memory length, number of tasks in the XLand task pool, single vs. multi-agent tasks, distillation teacher, and number of training steps. We use a consistent training configuration within each experimental comparison, but different configurations across different experimental comparisons. We therefore caution the reader against directly comparing results between different sections. For convenience, all experimental configurations are tabulated in Appendix G.

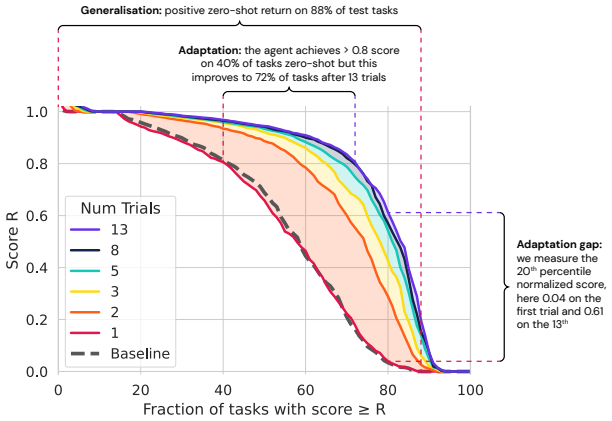


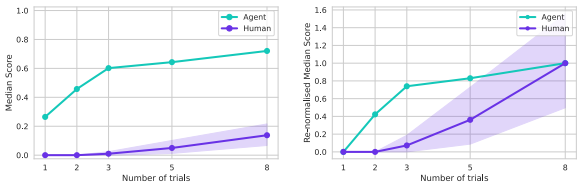
Figure 3: **Zero-shot generalisation and few-shot adaptation.** We plot the distribution of normalised task scores over the single-agent test set when evaluated with various numbers of trials. On the y-axis is the total last-trial reward relative to that of an agent fine-tuned on the test tasks (approximating "infinite trials" performance). Curves further towards the top right corner indicate better performance. When given more trials, the agent achieves higher scores in the last trial, showing test-time adaptation across most of the task distribution (shaded regions). The dashed line indicates the zero-shot performance of an agent trained in a regime where every episode consists of only a single trial.

3.1. AdA shows human-timescale adaptation

Single-agent. In Figure 3 we show the performance of AdA when trained in the single-agent setting described in Table G.1. Examine first AdA's zero-shot performance ($k = 1$, red line). This matches the performance of a baseline agent, which is trained identically to AdA, except that each episode consists of exactly one trial (i.e., with multi-task RL rather than meta-RL). In other words, AdA does not suffer any degradation in zero-shot performance, despite being trained on a distribution over number of trials $k \in \{1, 2, \dots 6\}$. Now turn your attention to AdA's few-shot performance ($k \in \{2, 3, 5, 8, 13\}$, orange to purple lines). Given more trials, AdA improves its performance on over 80% of the task set, clearly adapting at test time.

We compare the performance of AdA to that of a set of human players on 30 held-out hand-authored probe tasks, seeking to assess whether AdA adapts on the same timescale as humans. Figure 4a shows the median scores for AdA and for human players as a function of number of trials. Both AdA and human players were able to improve their score as they experienced more trials of the tasks, indicating that AdA exhibits human-timescale adaptation on this set of probe tasks. We provide details of the scores obtained on each task in Figure I.1. For full details of our human experiment design and ethics, see Appendix E.4.

Figure 5 analyses the behaviour of AdA in more detail on a specific held-out task. The increase in score with a larger number of trials indicates that the task is solved more consistently and more quickly when given a larger number of



(a) (b)

Figure 4: **Human-timescale adaptation.** We report median normalised last-trial score across 30 hand-authored tasks as a function of number of trials for AdA and human players. **(a)** shows the results using our standard per-task normalisation scheme. **(b)** re-normalises the results by subtracting the minimum and dividing by the maximum score per player-type, accounting for systematic differences between agents and humans. In particular, we see that AdA's re-normalised performance curve lies above that of humans, i.e, the timescale for improvement of AdA's score is at least as short as that for humans.

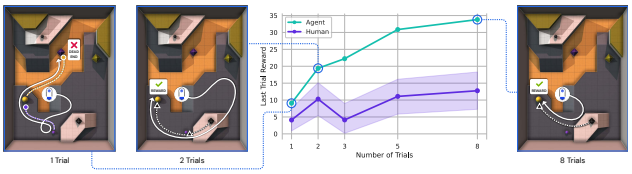


Figure 5: **Experimentation, success and refinement.** We report average performance and representative behaviour of AdA on the probe task `Wrong Pair Disappears` when evaluated with various numbers of trials. AdA's performance increases when given more trials, showing test-time adaptation. The top-down view images show representative last-trial trajectories when given different numbers of total trials.

trials. We can explain this effect in terms of the behaviour of AdA. When given 1 or 2 trials AdA shows structured hypothesis-driven exploration: trying out different combinations of objects and coming across the solution or a dead end. Once the solution is found, AdA refines its strategy on subsequent trials, gathering the correct objects with more efficiency and combining them in the right way. Thus AdA is able to generate a higher last-trial score when provided with more trials for refinement. We observe this pattern of behaviour consistently across many of our held-out probe tasks; See the following agent video and representative human trajectory. More videos are available on our microsite.

Multi-agent. We train a separate agent on a mixture of fully-cooperative multi-agent and single-agent tasks to explore adaptation in the multi-agent setting. In fully-cooperative multi-agent tasks, both players have the same goal. This gives rise to a variety of interesting strategic novelties that are absent in the purely single-agent setting, including division-of-labour and physical coordination. Here for the first time to our knowledge, we demonstrate that these behaviours can emerge at test time in few-shot on held-out tasks. Co-players for our training tasks are generated using fictitious self-play (Heinrich et al., 2015) and then curated using PLR, as in Samvelyan et al. (2022). For more details, see Table G.1 and Appendix G.4.

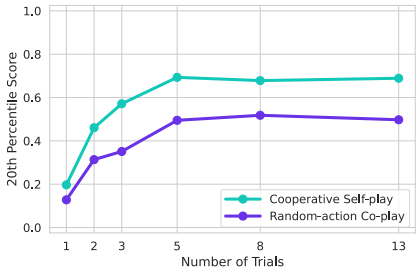


Figure 6: **Two heads are better than one.** Cooperative self-play outperforms single-agent performance on the test set of two-player cooperative held-out tasks. For this evaluation we restrict ourselves to tasks whose goals and production rules do not refer to players and which are solvable by a single player (216/1000 test tasks). To produce the purple curve, we evaluate AdA twice per task when playing with a random-action policy co-player, once playing as the first and once as the second player, and take the maximum score over both evaluations before cross-task aggregation. This accounts for possible advantages playing as one player might have over playing as the other in a task.

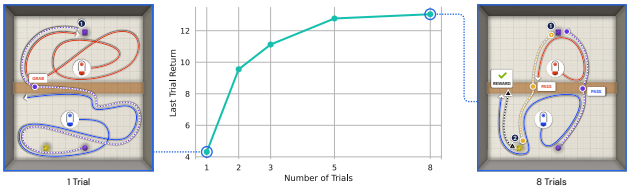


Figure 7: **Multi-agent coordination.** We report average performance and representative behaviour of AdA on the probe task `Pass Over the Wall Repeatedly` when evaluated in self-play with various numbers of trials. AdA's performance increases when given more trials, showing test-time adaptation. The top-down view images show representative last-trial trajectories when given different numbers of total trials.

Analogously with the single-agent setting, we find strong evidence of adaptation across almost 90% of the space of held-out test tasks (Figure H.1). Futhermore, we evaluate the resulting agent on a held-out test set of cooperative multi-agent tasks in two ways: in self-play and in co-play with a random-action policy. As shown in Figure 6, self-play outperforms co-play with a random-action policy by a large margin both in a zero-shot and in a few-shot setting. This indicates that the agents are dividing the labour required to solve the tasks, thereby solving the task more quickly (or at all) and improving their shared performance.

Examples of emergent social behaviour in self-play are shown in Figures 7 and H.2. Given only a few trials, the agents explore the space of possible solutions, sometimes operating independently and sometimes together. Given more trials, once the agents find a solution, they optimise their paths by coordinating physically and dividing labour to solve the task efficiently. This behaviour emerges from adaptation at test time and was not explicitly incentivised during training, other than through the fully cooperative reward function. We link a video of emergent multi-agent cooperation. Further videos are available on our microsite.

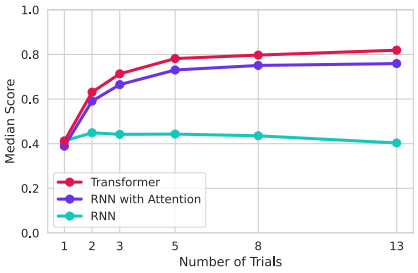


Figure 8: **Ablating architecture.** Adaptation over increasing numbers of trials for different choices of architectures. Incorporating attention modules is essential to achieve adaptation, with Transformer-XL architectures performing best.

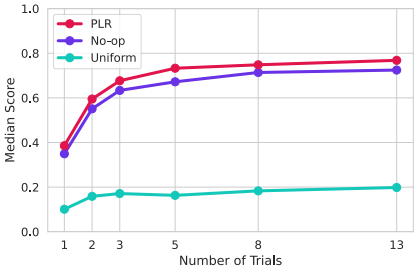


Figure 9: **Ablating curriculum.** Adaptation over increasing numbers of trials for different choices of curricula. No-op filtering and PLR greatly improve both zero-shot generalisation and few-shot adaptation over the uniform sampling baseline.

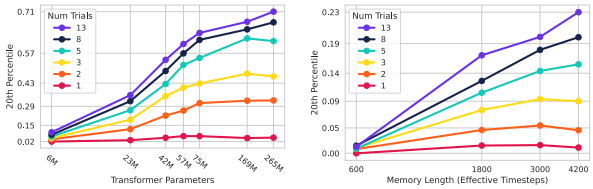
3.2. Architecture influences performance

We now dive deeper into understanding which components of our method are critical, via a series of ablation studies. In these studies we use a single initialisation seed, because we see low variance across seeds when training AdA (see Appendix I.3). All ablations are in the single-agent setting, unless stated otherwise.

First, we empirically contrast different choices of architectures: *Transformer-XL*, *RNN*, and *RNN with Attention*. To implement the RNN, we use a GRU (Cho et al., 2014). To facilitate comparison, we match the total network size for all architectures. Table G.4 shows details on the experimental setup. Figure 8 shows that while the Transformer-XL is the best performing architecture in this comparison, incorporating a multi-head attention module into an RNN recovers most of the performance of the Transformer, highlighting the effectiveness of attention modules.

3.3. Auto-curriculum learning improves performance

To establish the importance of automated curriculum learning, we compare adaptation when training with the curricula methods outlined in Section 2.2: no-op filtering and PLR. Figure 9 shows the median last-trial score of agents trained with different curricula. Both no-op filtering and PLR curricula strongly outperform a baseline trained with uniformly sampled tasks. Moreover, PLR outperforms no-op filtering, particularly at a higher number of trials, indi-



(a) (b)

Figure 10: **Scaling the model.** Plots show the 20$^{\text{th}}$ percentile test score for **(a)** Transformer parameters and **(b)** Transformer-XL memory length. Both axes are log-scaled, according to the functions $\log(x)$ and $-\log(1-y)$, respectively, and the relationship between model size and performance appears roughly linear on this scale. The slope is steeper when evaluating higher numbers of trials, showing that scaling the model is particularly effective at encouraging stronger adaptation, as opposed to stronger zero-shot generalisation.

cating that a regret-based curriculum is especially helpful for learning longer-term adaptation. In Appendix G.6 we detail training configuration, and also compare the sample efficiency of our methods, where we see that both auto-curriculum approaches are more sample-efficient than uniform sampling, in terms of both learning steps and FLOPs.

3.4. Scaling the agent increases performance

Methods that scale well are critical for continued progress in machine learning, and understanding how methods scale is important for deciding where to spend time and compute in the future. Scaling laws have been determined for many foundation models, where performance is related to model size and other factors as a power law, which can be seen as a linear relationship on a log-log plot. Inspired by such analyses, we investigate how adaptation scales with Transformer model size and memory length.

Scaling network size. We show how performance scales with the size of AdA's Transformer model, experimenting with the model sizes shown in Table G.9. When investigating scaling laws for model size, we follow Kaplan et al. (2020) in measuring only Transformer (i.e., non-embedding) parameters, which range across 3 orders of magnitude, from 6M to 265M Transformer parameters (i.e., from 41M to 533M total parameters). A complete list of hyperparameters is shown in Table G.10. Figure 10a shows that larger networks increase performance, especially when given more test-time trials to adapt. Model scale has particular impact on the lower percentiles of the test set. This indicates that larger models allow the agent to generalise its adaptation to a broader range of tasks. The roughly linear relationship between model size and performance on the log-log plot is indicative of a power law scaling relationship. That the curves are not exactly linear may be because we haven't trained to convergence, and because we use a 23M parameter distillation teacher in experiments for all model sizes. Appendix H.4 shows FLOPs adjusted results.

Scaling memory length. Adaptation performance also scales with the length of AdA's memory. The experimental setting is shown in Table G.11, where we examine the number of previous network activations we cache, investigating values from 100 to 700, which, with 6 Transformer-XL blocks, yields an effective timestep range of 600 to 4200 timesteps. Figure 10b shows that, as with model size, scaling memory length helps performance, especially in the lower test percentiles, pushing performance on the tails of the distribution. For any of our tasks, the maximum trial duration is 300 timesteps, so it is interesting that performance on, for example, 5 trials (1500 timesteps) continues to increase for "effective memory lengths" between 1800 and 4200. This indicates that it is easier for the Transformer-XL to make use of explicitly given memory activations rather than relying on theoretically longer-range information implicit in those activations.

3.5. Scaling the task pool increases performance

Another important factor to scale is the amount of data a model is trained on. For example, Hoffmann et al. (2022) showed that in order to get the most out of scaling a language model, one must scale the amount of training data at the same rate as the number of parameters. In our case, relevant data come from interaction with different tasks, so we examine the effect of scaling the number and complexity of different tasks in the XLand pool.

Scaling size of task pool. Here we examine the effect of varying the number of training tasks from which the auto-curriculum can sample. Table G.12 shows the full experimental setup for these comparisons. Figure H.4 shows higher test score for identically sized models on the larger task pool. As in the other scaling experiments, we especially see improved performance on the 20$^{\text{th}}$ percentile. We show results for two different sizes of models, with the larger Transformer yielding a larger gap when scaling the size of the task pool. This suggests the large models are especially prone to overfitting to a smaller task pool.

Scaling complexity of task pool. One final axis along which it is possible to scale our method is the overall complexity of the task distribution. For example, tasks with a flat terrain will be, on average, less complex to solve than tasks with terrain variation. In Figure H.5, we show that low environment complexity can be a bottleneck to scaling, by comparing the effectiveness of model scaling between agents trained on two distributions of the same size but different complexity and evaluated on their respective test sets. Open-ended settings with unbounded environment complexity, such as multi-agent systems, may therefore be particularly important for scaling up adaptive agents.

3.6. Distillation enables scaling agents

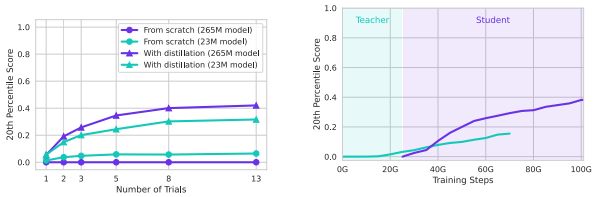
Now, we look at the role distillation plays in scaling. In short, we find that kickstarting training with a distillation period is crucial when scaling up model size. As shown in Figure 11a, training a 265M parameter Transformer model without distillation results in poor performance compared to a much smaller 23M parameter Transformer trained in the same way. However, when training with distillation from a 23M parameter teacher for the first 4 billion training frames, the 265M model clearly outperforms the 23M variant. For experiment details, see Appendix G.12.

Additionally, we find that even when the model size is the same for both student and teacher, we observe large gains from distillation, for a constant total frame budget (Figure 11). We speculate that this is due to bad representations learned early on by the student agent (Nikishin et al., 2022; Cetin et al., 2022), which can be avoided by using distillation. This is also consistent with findings in offline RL, where additional data is often required to effectively scale the model (Reid et al., 2022). The effect is largest for the first round of distillation, with diminishing returns in the next round of distillation (Figure H.5).

3.7. AdA leverages first-person prompting

Next, we prompt AdA with a first-person demonstration by a fine-tuned teacher. This process is analogous to prompting of large language models, where the agent's memory is primed with an example of desired behaviour from which it continues. AdA has never experienced a trajectory from a (human or agent) expert before during training. To prompt AdA, the teacher takes control of the avatar in the first trial, while AdA continues to receive observations as usual, conditioning its Transformer memory. AdA then proceeds on its own for the remaining trials and its scores are recorded in the usual manner. Figure H.11b compares prompted AdA with an unprompted baseline. Prompted AdA is unable to exactly mimic the teacher's demonstration in the second trial of a median task. It does, however, outperform an unprompted baseline across all numbers of trials, indicating that it is able to profitably incorporate information from the demonstration into its policy. We note that AdA was never trained with such off-policy first-person demonstrations, yet its in-context learning algorithm is still able to generalise to these. Thus meta-RL on a sufficiently vast and diverse array of tasks leads to unexpected and powerful emergent capabilities, a promising line for future research.

In Figure I.4 we provide prompting results for all single-agent hand-authored tasks and discuss the circumstances under which prompting is effective. In Appendix I.4 we also provide early results investigating prompting with human demonstrations on a subset of tasks. These reveal remarkable success in some cases, but also confirm that human demonstrations cannot overcome inherent limitations of AdA's task distribution. Linked videos compare AdA with and without without prompting.



(a) (b)

Figure 11: **Distillation improves performance.** **(a)** 20th percentile performance for adaptation over increasing numbers of trials when training from scratch or when kickstarting with distillation, for models with 23M and 265M Transformer parameters. Circle markers show training from scratch while triangle markers show training kickstarted with 4 billion frames of distillation. For this ablation, agents were trained in the multi-agent setup described in Section 3.1 and evaluated on the multi-agent test set after 22 billion total training frames. **(b)** Normalised last-trial score for $k = 13$ using the 23M parameter Transformer-XL. The teacher is trained from scratch, while the otherwise identical student is distilled from a snapshot of the teacher, taken after 25 billion steps of training.

4. Related Work

In this work, we leverage advances in attention-based models for meta-learning in an open-ended task space. Our agent learns a form of in-context RL algorithm, while also automatically curating the training task distribution; thus we combine two pillars of an *AI-generating algorithm* (Clune, 2019). The most similar work to ours is OEL Team et al. (2021), which also trains in a vast multi-agent task space with auto-curricula and generational learning. The key differences in our work are that we focus on *adaptation* (vs. zero-shot performance) and make use of large Transformer models.

Open-ended learning. Recent works have demonstrated the effectiveness of agent-environment co-adaptation with a distribution of tasks (Wang et al., 2019; Akkaya et al., 2019; Wang et al., 2020a; Parker-Holder et al., 2022). Our approach resembles the unsupervised environment design (Dennis et al., 2020)) paradigm, since we seek to train a generalist agent without knowledge of the test tasks. A pioneering method in this space was PAIRED (Dennis et al., 2020), which generates tasks using an RL-trained adversary. We build on Prioritised Level Replay (Jiang et al., 2021b;a), a later method which curates randomly sampled environments with high regret. Our work also relates to curriculum learning (Matiisen et al., 2020; Portelas et al., 2019; Sukhbaatar et al., 2018; OpenAI et al., 2021; Campero et al., 2021; Fang et al., 2021; Mu et al., 2022), with the key difference that these methods typically have a specific downstream goal or task in mind.

There have also been works that use auto-curricula over co-players, although these typically focus on singleton environments (Vinyals et al., 2019; Berner et al., 2019) or uniformly sampled tasks (Baker et al., 2020; Jaderberg et al., 2019; Liu et al., 2019; Cultural General Intelligence Team et al., 2022). Zhong et al. (2020) also train agents to generalise to unobserved environment dynamics, but they investigate zero-shot generalisation from language descriptions, whereas AdA discovers rules in few-shot at test time.

Adaptation. This work focuses on few-shot adaptation in control problems, commonly framed as *meta-RL*. We focus on *memory-based* meta-RL, building upon Duan et al. (2017) and Wang et al. (2016), who showed that if an agent observes rewards and terminations, and the memory does not reset, a memory-based policy can implement a learning algorithm. This has proven to be an effective approach that can learn Bayes-optimal strategies (Ortega et al., 2019; Mikulik et al., 2020) and may have neurological analogues (Wang et al., 2018). Our agents learn conceptual exploration strategies, something that would require the outer learner of a meta-gradient approach to estimate the return of the inner learner (Stadie et al., 2018). Our work is related to Alchemy (Wang et al., 2021), a meta-RL benchmark domain whose mechanics have inspired our production rules. The authors use memory-based meta-RL with a small Transformer, but find that the agent's performance is only marginally better than that of a random heuristic. Transformers have been shown to be effective for meta-RL on simple domains (Melo, 2022) and for learning RL algorithms (Laskin et al., 2022) from offline data. Adaptation also plays a critical role in robotics, with agents trained to adapt to varying terrain (Clavera et al., 2019; Kumar et al., 2021) or damaged joints (Cully et al., 2015).

Multi-agent. Fully cooperative multi-agent tasks typically have multiple Nash equilibria (Dafoe et al., 2020). When faced with a new problem, agents must adapt on-the-fly to agree on a single equilibrium of mutual benefit (Stone et al., 2010; Hu et al., 2020; Christianos et al., 2022). Division of labour and physical coordination have received extensive study in the multi-agent RL literature, but prior approaches have tended to focus on specific domains rather than training generalists, e.g. (Wang et al., 2020b; Yang et al., 2020; Strouse et al., 2021; Gronauer and Diepold, 2022).

Transformers in RL. Transformer architectures have recently shown to be highly effective for *offline* RL (Chen et al., 2021; Janner et al., 2021; Reed et al., 2022), yet successes in the *online* setting remain limited. One of the few works to successfully train Transformer-based policies was Parisotto et al. (2020), who introduced several heuristics to stabilise training in a simpler, smaller-scale setting. Indeed, while we make use of a similar Transformer-XL architecture (Vaswani et al., 2017; Dai et al., 2019), we demonstrate scaling laws for online meta-RL that resemble those seen in other communities, such as language (Devlin et al., 2019; Kaplan et al., 2020; Brown et al., 2020; Rae et al., 2021). Similarly, Melo (2022) use Transformers for fast adaptation in a smaller-scale meta-RL setting, interpreting the self-attention mechanism as a means of building an episodic memory from timestep embeddings.

5. Conclusion

The ability to adapt in minutes is a defining characteristic of human intelligence and an important milestone on the path towards general intelligence. Given any level of bounded rationality, there will be a space of tasks in which it is impossible for agents to succeed by just generalising their policy zero-shot, but where progress is possible if the agent is capable of very fast in-context learning from feedback. To be useful in the real world, and in interaction with humans, our artificial agents should be capable of fast and flexible adaptation given only a few interactions, and should continue to adapt as more data becomes available. Operationalising this notion of adaptation, we seek to train an agent that, given few episodes in an unseen environment at test time, can accomplish a task that requires trial-and-error exploration and can subsequently refine its solution towards optimal behaviour.

In this paper, we demonstrate, for the first time to our knowledge, an agent trained with RL that is capable of rapid in-context adaptation across a vast, open-ended task space, at a timescale that is similar to that of human players. This *Adaptive Agent* (AdA) explores held-out tasks in a structured way, refining its policy towards optimal behaviour given only a few interactions with the task. Further, AdA is amenable to contextual first-person prompting, strengthening its few-shot performance, analogous to prompting in large language models. AdA shows scaleable performance as a function of number of parameters, context length and richness of the training task distribution.

Our work highlights several crucial research directions for future progress towards increasingly general agents. Notably, we show it is possible to scale black-box meta-RL. We show that state-of-the-art auto-curriculum techniques can shape the data distribution to provide sufficient signal for learning to learn in an open-ended task space. Moreover, we demonstrate that attention-based architectures can take advantage of this signal much more effectively than purely recurrent networks, illustrating the importance of co-adapting data distribution and agent architecture for facilitating rapid adaptation. Finally, distillation enables us to realise the potential of large-scale Transformer architectures. We hope that our work will inspire progress in each of these areas, leading to increasingly capable and accessible foundation RL agents in the future.

References

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *CoRR*, abs/2108.13264, 2021.

I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

J. Albrecht, A. J. Fetterman, B. Fogelman, E. Kitanidis, B. Wróblewski, N. Seo, M. Rosenthal, M. Knutins, Z. Polizzi, J. B. Simon, and K. Qiu. Avalon: A benchmark for RL generalization using procedurally generated worlds. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2020.

D. Balduzzi, K. Tuyls, J. Perolat, and T. Graepel. Re-evaluating evaluation. *Advances in Neural Information Processing Systems*, 31, 2018.

C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.

V. Bhatt, B. Tjanaka, M. C. Fontaine, and S. Nikolaidis. Deep surrogate assisted generation of environments. In *Advances in Neural Information Processing Systems*, 2022.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. S. Chatterji, A. S. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

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. URL http://github.com/google/jax.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

A. Campero, R. Raileanu, H. Kuttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette. Learning with AMIGo: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations*, 2021.

M. Carroll, R. Shah, M. K. Ho, T. L. Griffiths, S. A. Seshia, P. Abbeel, and A. Dragan. On the utility of learning about humans for human-ai coordination, 2019.

E. Cetin, P. J. Ball, S. Roberts, and O. Celiktutan. Stabilizing off-policy deep reinforcement learning from pixels. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2784–2810. PMLR, 17–23 Jul 2022.

L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, 2021.

M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid, 2018.

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

F. Christianos, G. Papoudakis, and S. V. Albrecht. Pareto actor-critic for equilibrium selection in multi-agent reinforcement learning. *arXiv*, 2022. doi: 10.48550/ARXIV.2209.14344.

I. Clavera, A. Nagabandi, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019.

J. Clune. AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence. *CoRR*, abs/1905.10985, 2019.

K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. *CoRR*, abs/1812.02341, 2018.

K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2048–2056, 2020.

A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521:503–507, 2015.

Cultural General Intelligence Team, A. Bhoopchand, B. Brownfield, A. Collister, A. D. Lago, A. Edwards, R. Everett, A. Frechette, Y. G. Oliveira, E. Hughes, K. W. Mathewson, P. Mendolicchio, J. Pawar, M. Pislar, A. Platonov, E. Senter, S. Singh, A. Zacherl, and L. M. Zhang. Learning robust real-time cultural transmission without human data, 2022.

W. M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, and M. Jaderberg. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1331–1340. PMLR, 2019.

A. Dafoe, E. Hughes, Y. Bachrach, T. Collins, K. R. McKee, J. Z. Leibo, K. Larson, and T. Graepel. Open problems in cooperative AI. *CoRR*, abs/2012.08630, 2020.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285.

M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, and R. Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems*, 2022. doi: 10.48550/ARXIV.2206.06994.

M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019.

Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning, 2017.

L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

K. Fang, Y. Zhu, S. Savarese, and F.-F. Li. Adaptive procedural task generation for hard-exploration problems. In *International Conference on Learning Representations*, 2021.

G. Farquhar, K. Baumli, Z. Marinho, A. Filos, M. Hessel, H. P. van Hasselt, and D. Silver. Self-consistent models and values. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1111–1125, 2021.

A. Filos, E. Vértes, Z. Marinho, G. Farquhar, D. Borsa, A. L. Friesen, F. M. P. Behbahani, T. Schaul, A. Barreto, and S. Osindero. Model-value inconsistency as a signal for epistemic uncertainty. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 6474–6498. PMLR, 2022.

M. Fontaine, Y.-C. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis. On the importance of environments in human-robot coordination. 07 2021.

D. Grbic, R. Palm, E. Najarro, C. Glanois, and S. Risi. *EvoCraft: A New Challenge for Open-Endedness*, pages 325–340. 04 2021.

S. Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

D. Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *International conference on machine learning*, pages 805–813. PMLR, 2015.

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

M. Hessel, I. Danihelka, F. Viola, A. Guez, S. Schmitt, L. Sifre, T. Weber, D. Silver, and H. Van Hasselt. Muesli: Combining improvements in policy optimization. In *International Conference on Machine Learning*. PMLR, 2021.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

H. Hu, A. Peysakhovich, A. Lerer, and J. Foerster. "other-play"for zero-shot coordination. In *Proceedings of Machine Learning and Systems 2020*, pages 9396–9407, 2020.

M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

M. Jiang, M. Dennis, J. Parker-Holder, J. Foerster, E. Grefenstette, and T. Rocktäschel. Replay-guided adversarial environment design. In *Advances in Neural Information Processing Systems*, 2021a.

M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. In *The International Conference on Machine Learning*, 2021b.

M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. The Malmo platform for artificial intelligence experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016.

A. Juliani, A. Khalifa, V. Berges, J. Harper, E. Teng, H. Henry, A. Crespi, J. Togelius, and D. Lange. Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. In *IJCAI*, 2019.

N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi. Procedural level generation improves generality of deep reinforcement learning. *CoRR*, abs/1806.10729, 2018.

A. Kanervisto, S. Milani, K. Ramanauskas, N. Topin, Z. Lin, J. Li, J. Shi, D. Ye, Q. Fu, W. Yang, W. Hong, Z. Huang, H. Chen, G. Zeng, Y. Lin, V. Micheli, E. Alonso, F. Fleuret, A. Nikulin, Y. Belousov, O. Svidchenko, and A. Shpilman. Minerl diamond 2021 competition: Overview, results, and lessons learned, 2022.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel. A survey of generalisation in deep reinforcement learning. *CoRR*, abs/2111.09794, 2021.

A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

H. Küttler, N. Nardelli, A. H. Miller, R. Raileanu, M. Selvatici, E. Grefenstette, and T. Rocktäschel. The NetHack Learning Environment. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

M. Laskin, L. Wang, J. Oh, E. Parisotto, S. Spencer, R. Steigerwald, D. Strouse, S. Hansen, A. Filos, E. Brooks, M. Gazeau, H. Sahni, S. Singh, and V. Mnih. In-context reinforcement learning with algorithm distillation, 2022.

K.-H. Lee, O. Nachum, S. Yang, L. Lee, C. D. Freeman, S. Guadarrama, I. Fischer, W. Xu, E. Jang, H. Michalewski, and I. Mordatch. Multi-game decision transformers. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

S. Liu, G. Lever, N. Heess, J. Merel, S. Tunyasuvunakool, and T. Graepel. Emergent coordination through competition. In *International Conference on Learning Representations*, 2019.

D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*, page 185–201, Berlin, Heidelberg, 2018. Springer-Verlag. ISBN 978-3-030-01215-1.

T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher-student curriculum learning. *IEEE Trans. Neural Networks Learn. Syst.*, 31(9):3732–3740, 2020.

L. C. Melo. Transformers are meta-reinforcement learners. In *International Conference on Machine Learning*, pages 15340–15359. PMLR, 2022.

V. Mikulik, G. Delétang, T. McGrath, T. Genewein, M. Martic, S. Legg, and P. Ortega. Meta-trained agents implement bayes-optimal agents. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18691–18703. Curran Associates, Inc., 2020.

J. Mu, V. Zhong, R. Raileanu, M. Jiang, N. Goodman, T. Rocktäschel, and E. Grefenstette. Improving intrinsic exploration with language abstractions. In *Advances in Neural Information Processing Systems*, 2022.

R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, pages 16828–16847. PMLR, 2022.

OEL Team, A. Stooke, A. Mahajan, C. Barros, C. Deck, J. Bauer, J. Sygnowski, M. Trebacz, M. Jaderberg, M. Mathieu, N. McAleese, N. Bradley-Schmieg, N. Wong, N. Porcel, R. Raileanu, S. Hughes-Fitt, V. Dalibard, and W. M. Czarnecki. Open-ended learning leads to generally capable agents. *CoRR*, abs/2107.12808, 2021.

OpenAI, M. Plappert, R. Sampedro, T. Xu, I. Akkaya, V. Kosaraju, P. Welinder, R. D'Sa, A. Petron, H. P. de Oliveira Pinto, A. Paino, H. Noh, L. Weng, Q. Yuan, C. Chu, and W. Zaremba. Asymmetric self-play for automatic goal discovery in robotic manipulation, 2021.

P. A. Ortega, J. X. Wang, M. Rowland, T. Genewein, Z. Kurth-Nelson, R. Pascanu, N. Heess, J. Veness, A. Pritzel, P. Sprechmann, et al. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

K. Ota, D. K. Jha, and A. Kanezaki. Training larger networks for deep reinforcement learning, 2021.

E. Parisotto. *Meta Reinforcement Learning through Memory*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2021.

E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR, 2020.

J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. Foerster, E. Grefenstette, and T. Rocktäschel. Evolving curricula with regret-based environment design. In *The International Conference on Machine Learning*, 2022.

M. Pislar, D. Szepesvari, G. Ostrovski, D. L. Borsa, and T. Schaul. When should agents explore? In *International Conference on Learning Representations*, 2022.

R. Portelas, C. Colas, K. Hofmann, and P. Oudeyer. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 835–853. PMLR, 2019.

J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

R. Raileanu and T. Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*, 2020.

S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022.

M. Reid, Y. Yamada, and S. S. Gu. Can wikipedia help offline reinforcement learning? *CoRR*, 2022.

S. Risi and J. Togelius. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2, 08 2020. doi: 10.1038/s42256-020-0208-z.

M. Samvelyan, R. Kirk, V. Kurin, J. Parker-Holder, M. Jiang, E. Hambro, F. Petroni, H. Kuttler, E. Grefenstette, and T. Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

M. Samvelyan, A. Khan, M. D. Dennis, M. Jiang, J. Parker-Holder, J. N. Foerster, R. Raileanu, and T. Rocktäschel. MAESTRO: Open-ended environment design for multi-agent reinforcement learning. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *The International Conference on Learning Representations*, 2015.

J. Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463 vol.2, 1991. doi: 10.1109/IJCNN.1991.170605.

J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992. doi: 10.1162/neco.1992.4.2.234.

S. Schmitt, J. J. Hudson, A. Zidek, S. Osindero, C. Doersch, W. M. Czarnecki, J. Z. Leibo, H. Kuttler, A. Zisserman, K. Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

C. Schuhmann, R. Beaumont, R. Vencu, C. W. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. R. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In M. Fox and D. Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010.

D. Strouse, K. McKee, M. Botvinick, E. Hughes, and R. Everett. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34:14502–14515, 2021.

S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, 2017.

Y. Tay, M. Dehghani, S. Abnar, H. W. Chung, W. Fedus, J. Rao, S. Narang, V. Q. Tran, D. Yogatama, and D. Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling?, 2022.

J. Togelius and J. Schmidhuber. An experiment in automatic game design. In *2008 IEEE Symposium On Computational Intelligence and Games*, pages 111–118, 2008. doi: 10.1109/CIG.2008.5035629.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, . D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, Ç. Gülçehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782): 350–354, 2019. doi: 10.1038/s41586-019-1724-z.

L. Vygotsky. Interaction between learning and development. *Readings on the Development of Children*, pages 34–40, 1978.

J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

J. X. Wang, Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J. Z. Leibo, D. Hassabis, and M. Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868, 2018.

J. X. Wang, M. King, N. Porcel, Z. Kurth-Nelson, T. Zhu, C. Deck, P. Choy, M. Cassin, M. Reynolds, F. Song, et al. Alchemy: A structured task distribution for meta-reinforcement learning. *arxiv*, 2021.

R. Wang, J. Lehman, J. Clune, and K. O. Stanley. Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. *CoRR*, abs/1901.01753, 2019.

R. Wang, J. Lehman, A. Rawal, J. Zhi, Y. Li, J. Clune, and K. Stanley. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9940–9951. PMLR, 13–18 Jul 2020a.

T. Wang, H. Dong, V. Lesser, and C. Zhang. Roma: Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020b.

Z. Xu, J. Modayil, H. P. van Hasselt, A. Barreto, D. Silver, and T. Schaul. Natural value approximators: Learning when to trust past estimates. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

J. Yang, B. Petersen, H. Zha, and D. Faissol. Single episode policy transfer in reinforcement learning. In *International Conference on Learning Representations*, 2019.

J. Yang, A. Li, M. Farajtabar, P. Sunehag, E. Hughes, and H. Zha. Learning to incentivize other learning agents. In *Advances in Neural Information Processing Systems*. arXiv, 2020. doi: 10.48550/ARXIV.2006.06051.

W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2):2950–2957, 2020.

X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

V. Zhong, T. Rocktäschel, and E. Grefenstette. RTFM: generalising to new environment dynamics via reading. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

L. Zintgraf. *Fast adaptation via meta reinforcement learning*. PhD thesis, University of Oxford, 2022.

Appendix

A. Authors and Contributions

We list authors alphabetically by last name. Please direct all correspondence to Feryal Behbahani (feryal@google.com) and Edward Hughes (edwardhughes@google.com).

A.1. Full-time contributors

- **Jakob Bauer**: technical leadership, curriculum research, infrastructure engineering, task authoring, paper writing
- **Kate Baumli**: agent research, scaling, agent analysis, task authoring, paper writing
- **Feryal Behbahani**: research vision, team leadership, agent research, paper writing
- **Avishkar Bhoopchand**: technical leadership, evaluation research, infrastructure engineering, task authoring, paper writing
- **Michael Chang**: visualisation, agent analysis, human experiments
- **Adrian Collister**: XLand development, human experiments
- **Edward Hughes**: research vision, team leadership, evaluation research, paper writing
- **Sheleem Kashem**: infrastructure engineering, curriculum research, human experiments
- **Jack Parker-Holder**: curriculum research, paper writing
- **Yannick Schroecker**: agent research, scaling, task authoring, agent analysis, paper writing
- **Jakub Sygnowski**: infrastructure engineering, curriculum research, agent analysis, paper writing
- **Alexander Zacherl**: design leadership, agent analysis, task authoring, visualisation, human experiments
- **Lei Zhang**: curriculum research, agent analysis, paper writing

A.2. Part-time contributors

- **Nathalie Bradley-Schmieg**: project management
- **Natalie Clay**: QA testing, human experiments
- **Vibhavari Dasagi**: evaluation research
- **Lucy Gonzalez**: project management
- **Karol Gregor**: agent research
- **Maria Loks-Thompson**: XLand development, human experiments
- **Hannah Openshaw**: project management
- **Shreya Pathak**: agent analysis
- **Nicolas Perez-Nieves**: agent analysis, task authoring
- **Nemanja Rakicevic**: curriculum research, agent analysis
- **Tim Rocktäschel**: strategic advice, paper writing
- **Sarah York**: QA testing, human experiments

A.3. Advisers

- **Satinder Singh**: strategic advice
- **Karl Tuyls**: strategic advice

B. Acknowledgements

We thank Max Jaderberg for early guidance on the project vision. We are grateful to Wojciech Marian Czarnecki for an early version of the production rules formalism and Catarina Barros for a prototype implementation. We thank Dawid Górny for support on implementing visualisation tools. We are grateful to Alex Platonov for artistic rendering of the figures and accompanying videos. We thank Nathaniel Wong, Tom Hudson and the Worlds Team for their engineering support. Further, we thank Andrew Bolt, Max Cant, Valentin Dalibard, Richard Everett, Nik Hemmings, Shaobo Hou, Jony Hudson, Errol King, George-Cristian Muraru, Alexander Neitz, Valeria Oliveira, Doina Precup, Drew Purves, Daniel Tanis, Roma Patel, and Marcus Wainwright for useful discussions and support. We are grateful to Sebastian Flennerhag and Raia Hadsell for reviewing a draft of the paper.

C. Additional Related Work

Here we include additional related work related to procedural environment generation, which is a key component of our work.

Procedural environment generation. We make use of procedural content generation (PCG) to generate a vast, diverse task distribution. PCG has been studied for many years in the games community (Togelius and Schmidhuber, 2008; Risi and Togelius, 2020) and more recently has been used to create testbeds for RL agents (Justesen et al., 2018; Cobbe et al., 2018; Raileanu and Rocktäschel, 2020). Indeed, in the past few years a series of challenging PCG environments have been proposed (Juliani et al., 2019; Cobbe et al., 2020; Küttler et al., 2020; Samvelyan et al., 2021; Chevalier-Boisvert et al., 2018; Hafner, 2022; Deitke et al., 2022), mostly focusing on testing and improving generalisation in RL (Kirk et al., 2021; Bhatt et al., 2022; Fontaine et al., 2021). More recently there has been increased emphasis on open-ended worlds: Albrecht et al. (2022) proposed Avalon, a 3D world supporting complex tasks, while Minecraft (Johnson et al., 2016) has been proposed as a challenge for Open-Endedness and RL (Kanervisto et al., 2022; Fan et al., 2022; Grbic et al., 2021), but unlike XLand it does not admit control of the full simulation stack, thereby limiting the smoothness of the task space.

D. Environment Details

D.1. XLand 2.0

In this section we describe the differences between XLand 2.0 and the original XLand environment of (OEL Team et al., 2021). We modify the configuration space as follows:

- We introduce a new relation `touching(a,b)`. It is satisfied if objects `a` and `b` are in contact, as determined by Unity's collision detection with a distance threshold of 1 millimeter.
- We exclude all relations that refer to floors. As a more flexible alternative we introduce the option of spawning objects in a permanently frozen state, rendering them immobile. Frozen objects can be used as anchor points in the environment which require players to navigate to them by including them in goals or production rules.
- We only use predicates consisting of a single relation or its negation, excluding conjunctions or disjunctions of multiple relations. Note that production rules (Section 2.1) allow us to specify tasks which require the player to sequentially satisfy predicates, or to give them multiple ways to reach a desired state.

In addition we introduce the production rules system described in Section 2.1. There are three distinct mechanisms for hiding production rule information from the players:

1. Hiding a full production rule, where the player only gets information that a rule exists, but neither knows the condition nor what spawns.

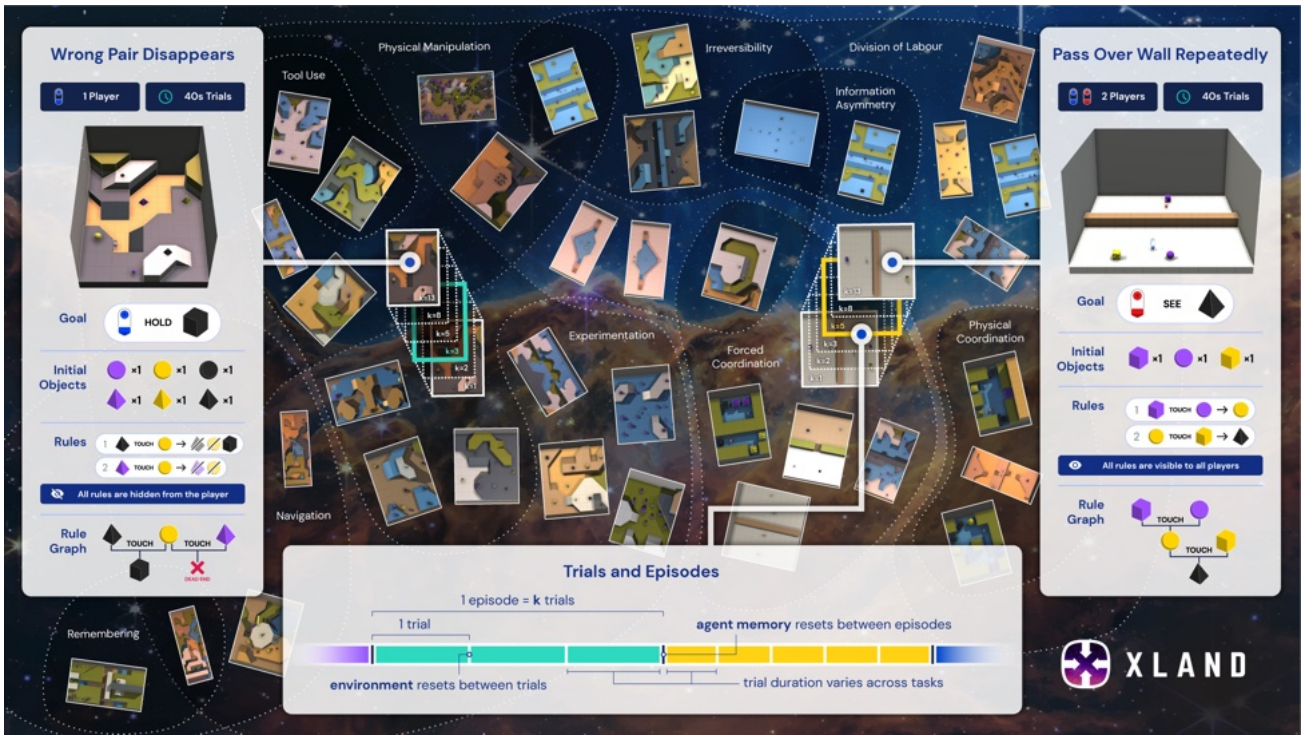


Figure D.1: **XLand 2.0: a vast, smooth and diverse task space of adaptation problems.** Different tasks have different adaptation requirements, such as experimentation, tool use or division of labour. For instance, in a task requiring experimentation, a player might be required to identify which objects can usefully combine, avoiding dead-ends, and then optimise the way in which they combine objects, like a toy version of experimental chemistry. Each task can be run for one or more trials, where the environment is reset between trials, but agent memory is not. Highlighted are two example tasks, `Wrong Pair Disappears` and `Pass Over Wall Repeatedly`, showing the goal, initial objects, production rules ("rules" in the figure) and how agents need to interact with them to solve the task. For full task descriptions see Appendix I.1.

2. Hiding an object, where a particular object is hidden from all production rules. The hidden objects are numbered such that if multiple objects are hidden, the agent can distinguish them.

3. Hiding a condition's predicate, where the agent gets to know the objects that need to satisfy *some* predicate, but it does not know which one. The hidden predicates are also numbered.

For example, consider the "Wrong pair disppears" probe task depicted in Figure E.1. Our probe task set (Table I.1) also contains an easier variant of this task "Wrong pair disppears, partial hiding". In this variant, only the input objects for each rule are hidden. In other words, the player is aware that one rule produces the desired `black cube` when two (unknown) objects touch, and that the other rule destroys the two input objects when two (unknown) objects touch. Moreover, they know that one object appears on the left-hand-side of both rules, because of the labelling. So the player can infer immediately that there exists a useful rule and a dead-end rule in this variant. In synthetic language, the rules are:

$$\texttt{touch}(?_1, ?_2) \rightarrow \texttt{black cube}$$
$$\texttt{touch}(?_3, ?_2) \rightarrow$$

Tables I.1 and I.2 contain further examples of partial hiding, including cases where information is hidden from one player but not the other, creating tasks with information asymmetry and thereby opportunities for social learning.

For the reader's convenience, Tables D.1 and D.2 respectively list all shapes and colours we used for objects in XLand. Table D.3 lists all predicates we used for goals and production rules.

Table D.1: Shapes used for objects.

Shape name
Wall
Cube
Sphere
Pyramid

Table D.2: Colours used for objects.

Colour name
Black
Purple
Yellow

Table D.3: Predicates used for goals and production rules.

Predicate name	Meaning
`touching(a,b)`	Whether `a` and `b` are in contact.
`near(a,b)`	Whether `a` and `b` are at most 1m apart.
`hold(a,b)`	Whether player `a` holds `b`.
`see(a,b)`	If `a` is a player, whether it can see `b`. If not, whether the line connecting the centres of mass of `a` and `b` is not obstructed by another object.
`not(p)`	Whether predicate `p` is not satisfied.

D.2. Pre-sampling tasks for training

The space of goals and production rules can generate at least 10^{40} distinct tasks, even given quite restrictive bounds on the number of objects and rules.[1] For efficiency and to reduce variance, we pre-sample a subset of this space using the method described below. In Section 3.5 we evaluate the effect of the size of the sampled set. For each task we sample a world using the procedure outlined in OEL Team et al. (2021) and combine it with a game and production rules as follows.

Single-player tasks. We start by uniformly sampling a player's goal, consisting of a predicate with optional negation and two objects. Then, for a fixed number of steps (which we sample uniformly between 1 and 4), we add new production rules, such that they need to be triggered in sequence to get the objects present in the goal. We initialise the world to contain the objects present in the condition of the first production rule, together with up to 10 *distractor* objects, not present in any production rule from this chain (nor in the goal).

Next, we introduce *dead-end* production rules. We sample them such that their condition contains either distractor objects or ones that are ultimately necessary for satisfaction of the goal, yet the spawns are always distractors. As such, triggering a dead end may put the game in an unsolvable state. Including them in the training games creates pressure on the agent to avoid indiscriminately triggering all production rules. Finally, we sample a hiding mask, specifying which part of the production rules will be visible or hidden from the player. The sampling of both the mechanism for hiding (described in Section 2.1) and the actual parts to hide is uniform random.

Multi-player tasks. For this work we restrict our multi-player games to fully cooperative two-player games. Such games are known to be particularly challenging, as they have multiple equilibria, and thus feature an equilibrium selection problem (Dafoe et al., 2020). To sample such a game, we start by sampling a single-player game as outlined above and randomly replace references to the first player in the goal or production rules with references to player two. We copy the goal and sample a new production rule hiding mask for player two, resulting in a fully cooperative two-player game with potentially asymmetric information provided about the task's production rules.

[1] This is an order of magnitude lower-bound estimate, assuming 4 shapes, 3, colours, 7 predicates, a maximum of 5 production rules with a maximum of 3 objects on the right-hand side, 7 blanking options, and a maximum of 20 objects in the scene.

E. Evaluation

E.1. Test scores

We evaluate our agents on a suite of 1000 held out test tasks sampled from the same distribution as the training games, using held-out world topologies. The procedure for pre-generating the XLand task pool is detailed in Appendix D.2. Rejection sampling ensures that no game (goal and production rules) in the test set is contained in the training set.

In XLand, rewards are obtained on every frame in which a goal is satisfied, making total rewards incomparable between tasks. To account for this, we fine-tune AdA on the test-task set. We compute the fine-tuned agent's maximum total last-trial reward (over any number of trials up to 13) and use this as an estimate of the maximum possible reward obtainable in a single trial of each test task. We call this quantity the normaliser. We define the *test score* S^i_m of an agent i on task m with k trials to be the total reward obtained in trial k divided by the normaliser. This normalises rewards roughly to the interval $[0, 1]$. Note that it is possible for an agent under evaluation to obtain a score greater than 1, both due to noise in the evaluation process and the fact that the agent under evaluation may be better than the one used for creating the normaliser.

When reporting the scores of our agents on a game with k trials, we always use the total reward of the last trial. This is a good measure of whether the agent has successfully navigated the exploration-exploitation tradeoff in a novel MDP, given knowledge of the number of trials k. If an agent is capable of adaptation, we expect to see the performance in the last (k^{th}) trial increase as a function of k: that is to say, the agent is able to make use of additional experience on-the-fly to perform better. We evaluate on $k \in \{1, 2, 3, 5, 8, 13\}$, where 8 and 13 are held out values of k that were not seen during training.

To aggregate the scores of an agent across games, we use a fixed (usually 20^{th}) percentile score (Agarwal et al., 2021). This gives us a lower bound guarantee on the performance of the agent across most of the tasks: for example, if the 20^{th} percentile score of an agent is 0.5, then the agent gets the score of at least 0.5 on 80% of the games. Using an aggregation method like this (as opposed to an average) allows us to concentrate on the coverage of many tasks, as opposed to focusing the effort on improving the performance on outlier tasks. Empirically, we find that our results are robust across a range of different percentiles.

E.2. Hand-authored probe tasks

Evaluation using test tasks can only give us information with respect to the pre-sampled distribution in Appendix D.2. While this is certainly vast and diverse, an arbitrary task sampled from the test set is not necessarily easily understandable for humans. So in addition to quantitative evaluation on 1000 test tasks we also investigate specific human-level capabilities of our agent on two sets of 30 single-agent and 28 multi-agent probe tasks. These are based on situations that are intuitive to humans and which require qualitatively different behaviours, which we can inspect in more detail "with the naked eye". A full description of all 58 probe tasks can be found in Appendix I. Representative single-agent and multi-agent probe tasks are described in detail in Figures E.1–E.4.

E.3. Adaptation metric

We introduce an *adaptation metric* to rank our agents based on their few-shot adaptation capability across the hand-authored probe task set. We collect last-trial total reward for all (m, k) pairs where m is a probe task and $k \in \{1, 2, 3, 5, 8, 13\}$. We normalise the per-task scores just as for the test set above. We then aggregate over tasks using the 50^{th} percentile (median). Finally, we aggregate over k by saying that agent A ranks higher than agent B if and only if A's task-aggregated scores are a Pareto improvement over B's. That is to say, we would like agents that are both capable of high-quality zero-shot generalisation where possible ($k = 1$), and also that can use additional trial information to efficiently improve their policy in few-shots $k > 1$; we don't "trade-off" between these.

A convenient way of using the Pareto improvement criterion to compute a scalar metric is the Nash average method of Balduzzi et al. (2018). We construct a competitive meta-game of "agents vs. k", and compute the maximum entropy Nash equilibrium for the game. The Nash payoff is then used as the adaptation metric, and agents are ranked by this metric. As desired, this metric has the property that if neither agent A nor agent B Pareto-dominate the other, the A and B receive the same Nash payoff and are therefore ranked equally. This adaptation metric was used as the means of selecting hyperparameters for training our best performing agent in Section 3.1.

Figure E.1: **Wrong Pair Disappears**: The player's goal is to hold a black cube, which does not exist among the initial objects. But there are two (hidden) production rules. The player needs to identify the correct world state which triggers the rule that creates the cube and not the one which destroys the necessary inputs. All this is embedded in a challenging world layout with one-way drops and limited visibility.

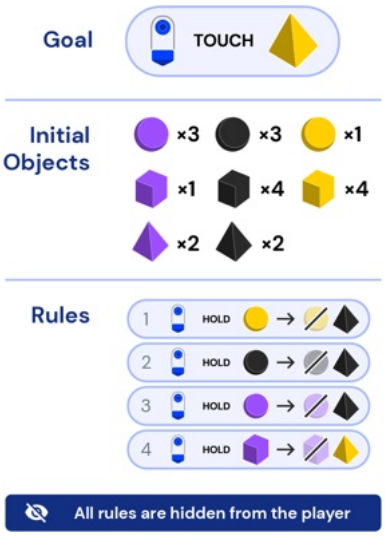


Figure E.2: **Pyramid in a Haystack**: To create the necessary yellow pyramid, the player needs to find and hold the purple cube. There are several distractor objects and distractor rules in this world, requiring the player to deal not just with a hard exploration challenge but also a very noisy environment.

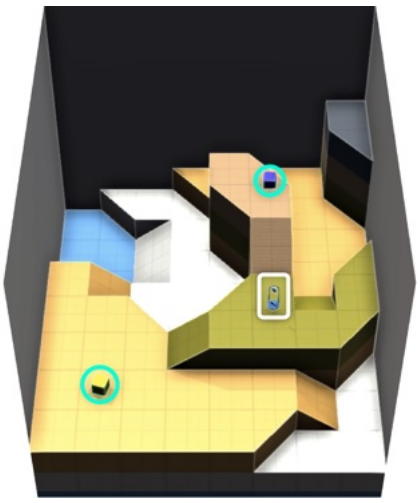
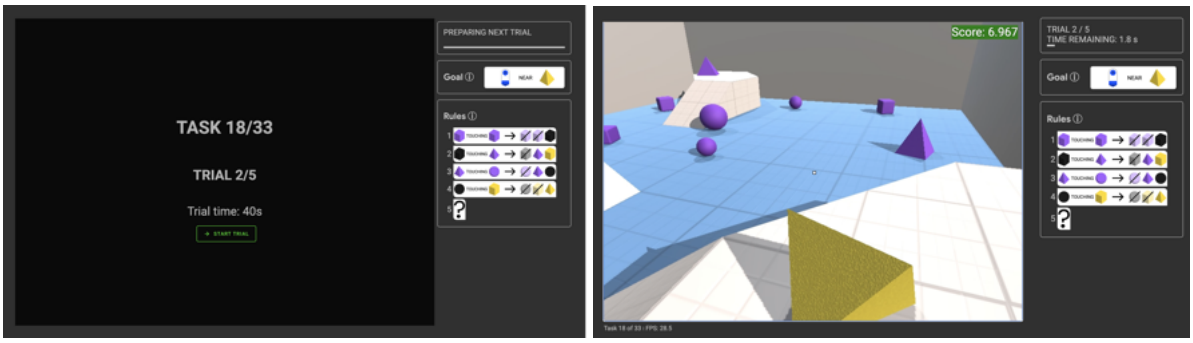


Figure E.3: **Push, don't lift**: The vast majority of training and evaluation tasks require lifting objects. Here two hidden rules destroy any object when lifted. In order to create the goal state, some "lateral thinking" is necessary: the player needs to identify that pushing the cubes with their body is possible.

Figure E.4: **Irreversible production for two**: Both players score when the first player holds a yellow sphere. There is no yellow sphere initially, but it can be produced from executing the first, second and fourth production rules in order. The other two rules are dead ends, destroying key input objects. Note that some input objects exist multiple times in the initial state, so there are multiple solution paths.



(a) (b)

Figure E.5: The human player interface. **(a)** Prior to each trial players are told how many trials they have remaining on the current task, and are given unlimited time to read the goal and production rules (subject to any hiding). **(b)** During the trial players observe the same first-person camera view as agents, but at a higher 800×600 pixel resolution. The goal, production rules, current score, and time remaining in the trial are displayed via UI elements.

E.4. Human data collection

To provide a benchmark for human-timescale adaptation, we collected score data from a pool of 100 human players on the 30 single-agent probe tasks. Before attempting the probe tasks, each player completed a graded training curriculum of 23 tasks to acquire familiarity with the mouse-and-keyboard control scheme and user interface (Figure E.5), and the particular game mechanics of XLand. Since humans cannot undergo a short-term memory "reset" on episode boundaries, individual players attempted each of the probe tasks for a single k only, with $k \in \{1, 2, 3, 5, 8\}$. All players experienced a variety of k across the task set. We assigned each player a unique ordering of the 30 probe tasks to average out any knowledge transfer from earlier tasks to later, and, within those orderings, we imposed separation between tasks with known similarity. Technical problems (e.g. internet dropout) prevented completion of 3.4% of the 3000 episodes, leaving an average of 19.3 samples per (task, k) pair, and a minimum of 17 samples for any individual (task, k).

Participants were recruited through an internal crowdsourcing pool. All participants provided informed consent prior to completing tasks and were reimbursed for their time. The consent form made participants aware that anonymous data on scores and gameplay would be made available as part of a future publication. The consent form also identified a potential risk, that playing the tasks might induce motion sickness. To mitigate this risk, participants were provided with regular opportunities to take breaks, and were free to withdraw from the study at any time. No PII was collected during the study. The full details of our study design, including compensation rates, were reviewed by our internal advisory group for human data before collection began.

XLand 2.0 is implemented in Unity, a state-of-the-art game engine that supports seamless real-time human play on a local PC. Due to technical requirements for our crowdsourcing pool, our participants interacted with XLand 2.0 not on a local PC but rather across a network connection to servers running the environment. Inevitably, interacting with the environment across a network has the potential to introduce some latency, due to uncontrollable factors in the network quality. While we did our best to mitigate any factors under our control (such as locating the servers physically close to the crowdsourcing pool), some participants still reported occasional lag.

F. Agent Details

In this work we relied in part on hyperparameter values that were tuned in previous literature, for example for some hyperparameters in the Muesli agent. For the remaining hyperparameters we performed either grid-search tuning (e.g. for the auto-curriculum) or tuned the parameters by hand over the course of several experiments (e.g. for the learning algorithm).

F.1. Learning algorithm

We use Muesli (Hessel et al., 2021) as our RL algorithm. We briefly describe the algorithm here, but refer the reader to the original publication for details. Taking a history-dependent encoding as input, in our case the output of an RNN or Transformer, AdA learns a sequence model (an LSTM) to predict the values $\hat{v}_i$, action-distributions $\hat{\pi}_i$ and rewards $\hat{r}_i$ for the next I steps. Here, $i = 0, \ldots, I$ denotes the prediction i steps ahead. I is typically small and in our case $I = 4$. For each observed step t, the model is unrolled for I steps and updated towards respective targets:

$$\mathcal{L}_r^t = \sum_{i=0}^{I} \left(\hat{r}_i^t - r_{t+i}\right)^2, \tag{1}$$

$$\mathcal{L}_v^t = \sum_{i=0}^{I} \left(\hat{v}_i^t - G_{t+i}\right)^2, \tag{2}$$

$$\mathcal{L}_\pi^t = \sum_{i=0}^{I} \mathrm{KL}\left(\pi_{\mathrm{CMPO}}^{t+i} \,||\, \hat{\pi}_i^t\right). \tag{3}$$

Here, r_{t+i} refers to the observed rewards. G_{t+i} refers to value-targets which are obtained using Retrace (Munos et al., 2016) based on Q-values obtained from one-step predictions of the model.

The action-targets π_{CMPO}^t are obtained by re-weighting the current policy[2] using clipped, normalised, exponentially transformed advantages. Muesli furthermore incorporates an additional auxiliary policy-gradient loss based on these advantages to help optimise immediate predictions of action-probabilities. Finally, Muesli maintains a target network which trails the sequence model and is used for acting and to compute Retrace targets and advantages. Table F.1 details the full learning algorithm hyperparameters. We refer the reader to the Muesli paper for detailed explanations about what each parameter value is for.

F.2. Agent Architecture

Here we provide an overview of the agent architecture, and provide important hyperparameters for the network architecture.

Observation encoder. The first-person view RGB observation (Table F.2) is passed through a ResNet (He et al., 2016) with $[16, 32, 32]$ convolutional channels, each consisting of 2×2 blocks and a final output size of 256. Max-pooling is used, in addition to scalar residual multipliers. `relu` activations are used throughout.

The goal observation is passed through a goal embedder, which is the same in OEL Team et al. (2021). This maps each of the 6 goal elements (negation, predicate, shape of object 1, colour of object 1, shape of object 2, colour of object 2) in the goal representation to a dense embedding vector of size 8. These are concatenated together and passed through a 3-layer MLP of size $[8, 8, 8]$, resulting in a final goal embedding of size 8.

[2]The prior distribution is actually a mixture of the current estimate of the policy, the (outdated) policy used to produce the sample and the uniform distribution where the latter two are mixed in as regularisers.

Table F.1: Learning algorithm hyperparameters.

Hyperparameter	Value
Model rollout steps	4
Search params update rate	0.001
Value loss weight	1.0
Policy loss weight	3.0
Reward loss weight	1.0
CMPO Policy loss weight	1.0
Retrace λ	0.95
Optimizer	Adam
Adam - max absolute update	1.0
Adam - epsilon	1e-8
Adam - learning rate	1e-4 (fixed schedule)
Discount	0.99
Batch size	144 sequences
Sequence length	80 frames
Actors per learner	12,000
CMPO loss p(uniform prior)	0.003
CMPO loss p(actor prior)	0.03

Production rules are encoded in the same manner as the goal, but mapped through a larger final MLP of shape $[512, 256]$ resulting in a final production rule embedding vector of size 256.

The encoded RGB, goal, and production rules observations are concatenated together with all remaining scalar and vector observations (including previous reward, previous action, proprioception observations, and trial timing information) and passed though a final MLP. This results in an encoded observation vector of a size matching the hidden dimension of the Transformer memory.

Transformer memory. We use a Transformer-XL with causal masking (Dai et al., 2019). For the actor step we use a context window of 1, and in the learner step we use a rollout context window of 80. The Transformer-XL memory uses 300 previous cached activations (1800 effective timesteps) of memory in all experiments unless otherwise stated. We apply layer normalisation before attention operations as in Parisotto et al. (2020), use gating in the feedforward component as in Shazeer (2020), and apply relative positional embeddings as in Dai et al. (2019). As is common with Transformers, we use the `gelu` activation function throughout (Hendrycks and Gimpel, 2016).

Transformer-XL enables the use of longer, variable-length context windows by concatenating a cached memory of previous attention layer inputs to the keys and values during each forward pass. Since inputs to intermediate layers are activations from the previous layer, which in themselves contain information about the past, caching M activations theoretically allows for an effective memory horizon of $M \times L$, where L is the number of attention layers in the network.

Muesli sequence model and prediction heads. Next, we take the Transformer-XL output embedding, and use two MLPs of width 1000 to produce the hidden and cell values for the initial state of the Muesli model LSTM. On top of the hidden value, we apply MLP heads of width 1000 for the policy and value respectively. The policy MLP is followed by 6 linear softmaxed outputs corresponding to 6 action groups in a decomposed action space for the policy (as in OEL Team et al. (2021)). The value MLP is followed by a 601-unit binned logit prediction as in Hessel et al. (2021). Finally, the Muesli sequence model is unrolled for a further 4 steps, starting from the LSTM state embedding and producing a 1000-dimensional output vector on each LSTM step. This feeds into a further 1000-dimensional MLP followed by a 601-unit binned reward prediction.

F.3. Observations

We summarise all the observations received by the agent when running inference in Table F.2. In the descriptions, "legacy reasons" refers to an observation format that was inherited from OEL Team et al. (2021).

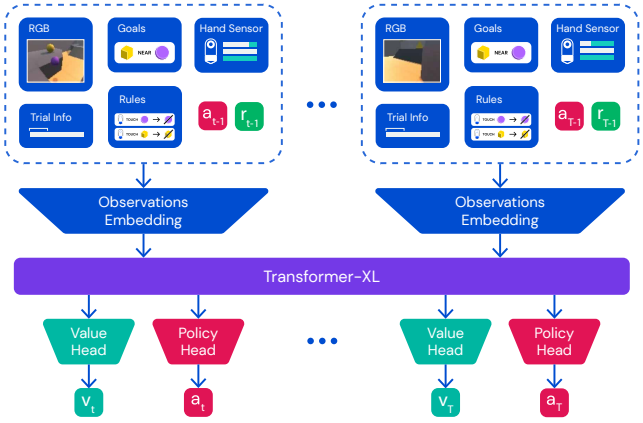


Figure F.1: **Agent architecture.** For each timestep, we embed and combine the pixel observation, goal, hand, trial and time information, production rules, previous action, and previous reward into a single vector. These observations embeddings pass in sequence to the Transformer-XL, whose output embeddings feed into an MLP value head, MLP policy head, and the Muesli LSTM model step (omitted in the diagram for brevity). See Appendix F.2 for more details about our agent architecture.

Table F.2: Agent observations.

Observation name	**Shape**	**Meaning**
RGB	$72 \times 96 \times 3$	RGB values of the first-person view of the agent.
IS HOLDING	1	Integer in $\{0, 1\}$ indicating whether the agent is holding an object.
HAND DISTANCE	1	Distance to the held object as a fraction of the agent's maximum reach (0 while no object is held).
HAND FORCE	1	The force exerted by the agent on the held object as a fraction of its maximum grip force (0 while no object is held).
LAST ACTION	10	Last action performed by the agent.
GOAL ATOMS	6×6	Agent-readable description of the goal. For legacy reasons, only the first row is non-zero. The first row (6 numbers) describes the goal with elements: 1: whether the goal is negated or not, 2: index of the binary predicate, 3-4: shape and colour of the first objects, 5-6: shape and colour of the second object.
GOAL SOP MATRIX	6×6	Always the same, kept for legacy reasons.
ATOM REWARDS	6	The reward of the agent in the previous frame (1 number), padded with zeros for legacy reasons.
OPTION REWARDS	6	Same as ATOM REWARDS, kept for legacy reasons.
PRODUCTION RULES	16×26	A description of up to 16 production rules. A single production rule is described as: $3 \times 6 = 18$ numbers describing up to three object-predicate-object triggers and $4 \times 2 = 8$ numbers describing up to 4 spawn objects. We only ever use a single object-predicate-object trigger for all tasks in this paper. Hiding (Section 2.1) is implemented by adding extra predicate and shape indices meaning hidden production rule, first hidden object, etc.
REWARD	1	The environment reward obtained in the previous step.
TRIALS REMAINING	5	One-hot encoding of the number of trials remaining in the episode, up to a maximum of 5 trials remaining.
MORE THAN 5 TRIALS	1	Integer in $\{0, 1\}$ indicating whether there are more than 5 trials remaining in the episode.

TIME UNTIL LAST TRIAL	1	Time remaining (in seconds) until the final trial in the current episode.
TIME LEFT IN CURRENT TRIAL	1	Time remaining (in seconds) until the end of the current trial.
DURATION OF LAST TRIAL	1	The duration (in seconds) of the final trial in this episode. In our setting, all trials for the same task have the same duration.
DURATION OF NEXT TRIAL	1	The duration (in seconds) of the next trial in the current episode. In our setting, all trials for the same task have the same duration.

G. Training Details

G.1. Meta-RL

AdA is trained with a meta-RL setup, in which episodes of experience for the agent comprise multiple trials of interaction with the task environment, where the task is reset on trial boundaries. In this setting, it is known that the agent's policy can converge to Bayes-optimal behaviour, experimenting on-the-fly to reduce its epistemic uncertainty, and reusing discovered information to achieve goals increasingly efficiently. It is important that the agent's memory is not reset at trial boundaries, but only at episode boundaries. Similarly, the agent trains with a fixed discount factor $\gamma \neq 0$ throughout the episode, including on trial boundaries. For training, we sample (m, k) pairs according to a factorised distribution $(\rho_{\mathcal{M}}, \rho_K)$, the parameters of which are controlled by an automatic curriculum (Section 2.2). The MDPs m are all drawn from a procedurally generated domain $\mathcal{M}$, called XLand (Section 2.1). We choose as our space of trials $K = \{1, 2, \dots 6\}$, and provide our agent with k as a conditioning input. After training, our agent is capable of few-shot adaptation across a wide-range of MDPs, including in held-out $m \in \mathcal{M}$ on which $\rho_{\mathcal{M}}$ puts no probability mass, and when $k > 6$.

G.2. Distillation

To integrate the distillation loss with Muesli, we unroll the model from every transition observed by the student. We minimise the KL-divergence between all of the action-probabilities predicted by the model and the action-probabilities predicted by the teacher's policy at the corresponding timestep. Analogously to Muesli's policy-loss $\mathcal{L}_\pi$ defined in (3), we define

$$\mathcal{L}_{\text{dist}} = \sum_{i=0}^{I} \text{KL}\left(\tilde{\pi}_0^{t+i} \;||\; \hat{\pi}_i^t\right), \tag{4}$$

where $\tilde{\pi}$ corresponds to the predicted action-logits provided by the teacher given the same observed history. We weight the distillation loss with 4.0 and unless otherwise stated use this loss for only the first 4B frames of training. Furthermore, we found it useful to add additional L^2 regularisation during distillation with a loss weight of 1e-6.

G.3. Single-agent training

Our single-agent training setup uses a task pool generated as described in Section 2.1. The experimental setup for the single-agent distillation teacher is summarised in Table G.2. Single-agent training used an earlier version of XLand 2.0 than multi-agent experiments, without the frozen objects described in Section D.1. Frozen objects were also therefore excluded from the test and hand-authored probe task sets. AdA was implemented using JAX (Bradbury et al., 2018) and trained on 64 TPU devices. The wall-clock time for training this version of AdA from scratch was approximately 5 weeks: 1 week to train the teacher, and 4 weeks to train AdA. Even after this amount of training, AdA had not reached convergence, illustrating the benefits of open-ended learning methods.

Table G.1: Experimental setup for experiments in Section 3.1.

# players	Model parameters	Memory	Task pool	Curriculum	Teacher	Steps
1	169M TXL / 353M total	1800	25B	PLR G.6	G.2	100B
2	265M TXL / 533M total	1800	see G.4	PLR G.6	G.3	70B

In Section 3 we report normalised scores for AdA. The normalisation factor is the total per-trial reward achieved by a reference agent, which has been fine-tuned to approximate "infinite trials" performance. Here we describe the fine-tuning

procedure used to create this reference agent. We start with an AdA agent (an earlier checkpoint of our final agent at 40G steps) and load it into a new training experiment. We replace the usual training tasks and curriculum with the test or hand-authored tasks. We remove the auto-curriculum and just do uniform sampling over the tasks and k. The memory reset scheme is the same as described in Section 2.3. We train this agent until it converges on the given task set. It took less than 0.5G steps to converge. Now the agent should have something approximating an optimal policy on each task. We took the maximum score over all k (usually $k = 13$) as the normaliser for that task.

Table G.2: Distillation teacher for the single-agent experiments in Section 3.1.

Model Parameters	Memory	Task pool	Curriculum	Teacher	Steps
23M TXL / 76M total	1800	25B	PLR G.6	None	25B

G.4. Multi-agent training

Starting from the 25B-sized task pool used for single-agent training, we generate a two-player task pool of the same size by the procedure described in Appendix D.2. For all our multi-agent experiments, we use a half-half mixture of single-player and two-player tasks, as described in Section 2.1. For each task we decide whether to spawn some of the initial objects permanently frozen (see Appendix D.1) with 50% probability. For tasks with frozen objects, we iterate over the initially spawned object types and freeze all spawned instances of this type with a probability of 20%, while ensuring that the task remains solvable.

During training we uniformly sample a co-player policy from a pool generated using fictitious self-play. The co-player pool is initialised with a random-action policy. Every 500M training frames we add a snapshot of the learning agent to the pool, thereby adding more and more capable co-players over time. Finally we apply the PLR auto-curriculum method (see Section 2.2) to curate the tasks (worlds, games and co-players) using the agent's TD-error based fitness. The experimental setup for the multi-agent distillation teacher is summarised in Table G.3.

Table G.3: Distillation teacher for the multi-agent experiments in Section 3.1.

Model Parameters	Memory	Task pool	Curriculum	Teacher	Steps
23M TXL / 76M total	1800	see Sec G.4	PLR G.6	None	22B

G.5. Architecture experiments

Table G.4 shows the experimental setup for the experiments comparing different memory architectures in Section 3.2.

Table G.4: Experimental setup for comparing different memory architectures.

Architecture	Parameters	Memory	Task pool	Curriculum	Teacher	Steps
Transformer-XL	76M total	1800	25B	No-op	None	50B
GRU with Attention		-				
GRU		-				

G.6. Auto-curriculum learning

No-op filtering details. Here we provide additional details of No-op filtering. For each task from the XLand training pool, we evaluate the learning agent (without sending any experience to the learner) and no-op policy on the task for 10 independent episodes, each of length 1 trial, producing scores $\{R_0, \ldots, R_9\}$, $\{R'_0, \ldots, R'_9\}$, respectively. We admit the proposal task for training if it satisfies the following criteria:

1. $\max R'_i \leq \epsilon_1$ (No-op is not too good.)
2. $|\{i : R_i \geq \epsilon_2\}| \leq \epsilon_3$ (Agent is not too good.)

3. $|\{i : R_i \geq \max R'_i + \epsilon_0\}| \geq \epsilon_4$ or $|\{i : R_i \leq \min R'_i - \epsilon_0\}| \geq \epsilon_5$ (Agent is sufficiently different from no-op.)
4. $\max R_i - \min R_i \geq \epsilon_6$ (Agent scores have sufficient variance.)

The ϵ_i's are thresholds and become hyperparameters in our training setup. Since different tasks have different durations in general, we use relative thresholds defined as a fraction of trial duration for ϵ_0, ϵ_1, ϵ_2, ϵ_6 and absolute thresholds for the rest. Once a task is admitted for training, it is run using the full number of trials specified by the task and for 30 episodes. All experience from these runs are sent to the learner. See Table G.5 for the hyperparameters used.

Table G.5: No-op filtering hyperparameters.

Parameter	Value	Relative to trial duration
ϵ_0	0.01	Y
ϵ_1	1.1	Y
ϵ_2	0.4	Y
ϵ_3	5	N
ϵ_4	1	N
ϵ_5	3	N
ϵ_6	0.01	Y

PLR details. Here we provide additional details for Prioritised Level Replay (PLR), used in training AdA. PLR uses a *fitness score* that approximates the agent regret for a given task (Jiang et al., 2021a;b). PLR maintains an archive $\mathcal{P}$ of tasks to replay with fixed maximum size. With probability p (referred to as the *replay probability*) a task is sampled from $\mathcal{P}$ while taking into account the fitness score and staleness of each task (see Jiang et al. (2021b), Section 3) to train the learning agent. The staleness represents how much time has passed since the task was last sampled, and ensures that all tasks in $\mathcal{P}$ have accurate scores. The final probabilities are computed by combining the fitness and staleness scores, with staleness weighted using the parameter $s \in [0, 1]$.

Tasks are added to $\mathcal{P}$ by first sampling with probability $1 - p$ a proposal task from the training task set and evaluating its fitness score. If the fitness score is greater than the minimum fitness score of all tasks in $\mathcal{P}$, the proposal task is added to $\mathcal{P}$. If the new size of $\mathcal{P}$ exceeds the fixed maximum size, the lowest fitness task is dropped. Note that in PLR, unlike in No-op filtering, a task can potentially be trained on indefinitely, if it never leaves $\mathcal{P}$.

We found that using last-trial fitness led to better empirical performance and sample efficiency than first or average trial fitness. This is likely because in earlier trials, error-based fitness is higher as the agent is pursuing exploratory behavior, which should not be taken as a sign that the policy is sub-optimal. However, high error-based fitness in the last trial likely indicates a sub-optimal policy when solving the task after time for adaptation, analogous to the regret approximation in the original single-trial PLR.

In order to use the last-trial fitness as our objective we need to make a number of changes to the original PLR framework, which was designed for single trials in a more homogeneous domain. We denote the per-step fitness score at the i^{th} step of trial k by $f_{i,k}$. First, to avoid adding a bias towards longer trial durations we use the *average* per-trial fitness score $\tilde{f}_k \sum_i \tilde{f}_{i,k}/N_k$ where N_k is the number of steps per trial. Next, to ensure we do not prioritise lower values of k, which tend to have a higher average last-trial fitness score, we then normalise $\tilde{f}_k$ by $f_k(\tilde{f}_k - \mu_k)/\sigma_k$ where μ_k and σ_k are rolling per-trial means and variances for each trial index, calculated from all evaluated tasks. Finally, we can define the fitness for PLR to be f_k, the normalised last-trial fitness score.

As described in Jiang et al. (2021a;b), PLR contains the following hyper-parameters: replay probability p, maximum replay size N_{max}, minimum replay size N_{min} (we set $p = 0$ if $|\mathcal{P}| < N_{\text{min}}$), N_{train} the total number of trials for which to run a training task before re-sampling, and s the staleness coefficient. See Table G.6 for the hyper-parameters used. We conducted a grid search over the replay probability $p \in \{0.1, 0.2, 0.5\}$, size of the replay pool $N_{\text{max}} \in \{1000, 10000, 50000\}$ and staleness coefficient $s \in \{0.1, 0.2\}$, and in all cases set N_{min} to be 90% of N_{max}.

PLR fitness metric. It remains for us to define the per-step fitness score $f_{i,k}$. For this, we use the simplest regret-like metric, the 1-step TD-error (Jiang et al., 2021b; Schaul et al., 2015). Concretely, we estimate the *TD-error fitness* based on the immediate value-predictions of the model: $|r_t + \gamma \hat{v}_0^{t+1} - \hat{v}_0^t|$. In some settings this may be undesirable, for example,

Table G.6: PLR hyperparameters.

Parameter	Value
p	0.2
N_{max}	1000
N_{min}	900
N_{train}	30
s	0.2

TD-errors typically increase as the agent achieves higher rewards. Therefore, we also propose to compute fitness metrics based on the Muesli dynamics model. Rather than simply using the accuracy of the model prediction, we look at the impact of the prediction on the value function and action predictions. We define the *value-model fitness* as $|\hat{v}_0^{t+1} - \hat{v}_1^t|$, the difference between the value estimate at the predicted next state and the true next state. We also define a value-agnostic metric, the *action-model fitness* as follows: $JS(\hat{\pi}_0^{t+1}, \hat{\pi}_1^t)$, i.e. the difference between the action predictions at the predicted next state and the actual next state, where difference is measured with the Jensen-Shannon divergence (Xu et al., 2017; Farquhar et al., 2021; Filos et al., 2022; Pislar et al., 2022).

In Figure G.1 we show training curves for TD-error fitness, value-model fitness, and action-model fitness. Table G.7 shows the experimental setup for these experiments. We see that both TD-error and action-model fitness metrics outperform the value-model fitness. We chose TD-error for our PLR training runs because it has better asymptotic performance in both the zero-shot and the few-shot setting, and because it was shown to perform well in previous work (Jiang et al., 2021a).

Table G.7: Experimental setup for comparing different PLR fitness functions.

Model parameters	Memory	Task pool	Curriculum	Teacher	Steps	Fitness function
23M TXL / 76M total	1800	25B	PLR	None	25B	TD error Value model Action model

Curriculum efficiency. Next, we compare training sample efficiency for baseline uniform sampling and the different curriculum methods, in units of both learner steps and FLOPS. Figure G.2 shows the median last-trial scores for few-shot ($k = 13$) and zero-shot evaluation tasks as a function of learner steps. We see that both No-op filtering and PLR curricula

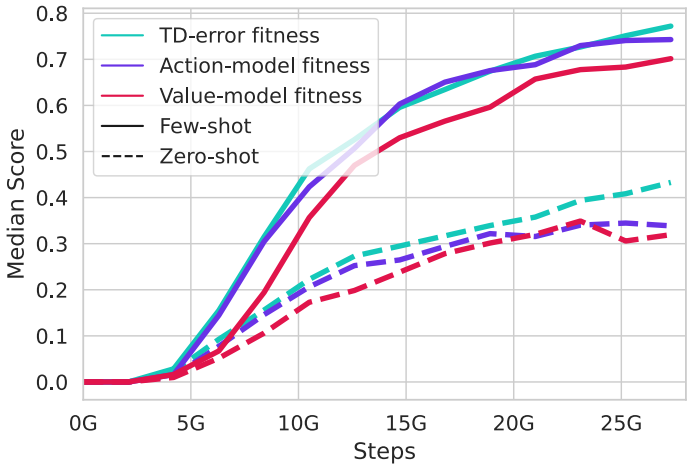
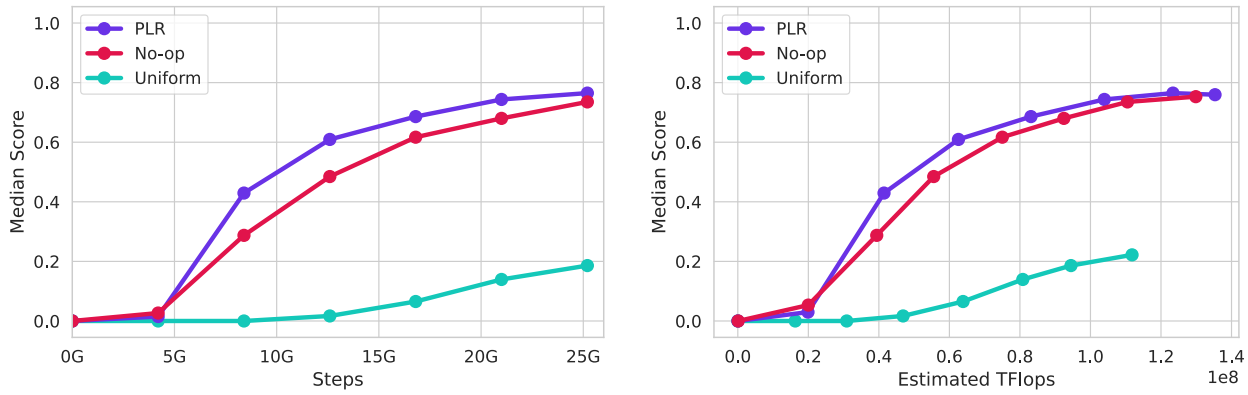


Figure G.1: PLR fitness metric comparison for zero-shot generalisation ($k = 1$) and few-shot adaptation ($k = 13$). We compare the TD-error fitness used in our main agents against two approaches using the Muesli dynamics model. We see that action-model fitness matches TD-error fitness in few-shot performance, with weaker zero-shot performance.



(a) (b)

Figure G.2: **(a)** Sample efficiency in steps for different choices of curricula. Both No-op and PLR significantly improves sample efficiency over uniform sampling of tasks. Few-shot denotes $k = 13$ score and zero-shot denotes $k = 1$ score. **(b)** Sample efficiency in FLOPs for different choices of curricula. No-op has an initial advantage over PLR, but PLR outperforms No-op later in training.

strongly improve training sample efficiency over uniform sampling, with PLR being more efficient early in the training process. When we plot the same few-shot median performance as a function of FLOPS, we see that No-op has a slight early advantage, but PLR outperforms No-op later in training. The initial advantage for No-op may be because No-op expends more FLOPS (10 evaluations per task vs. 1 in PLR) for task evaluation, which finds higher quality training tasks at the start of training.

Emergent curricula. In Figure G.3 we show task metrics analysing the tasks selected by PLR and No-op filtering. Neither method optimises for these metrics, hence their apparently curriculum (from low values to higher ones over time) is "emergent".

G.7. Distillation teacher for scaling experiments

In all of our scaling experiments (Sections 3.4 and 3.5), we distill the policy from an identical teacher snapshot to ensure our experiments are comparable. Training details for the teacher are detailed in Table G.8. This teacher is used to kickstart our agents for their first 4B frames of training.

Table G.8: Distillation teacher for scaling experiments.

Model parameters	Memory	Task pool	Curriculum	Teacher	Steps
23M TXL / 76M total	1800	200M	No-op	None	23B

G.8. Scaling the network

Table G.9 shows the experimental setup for the model size scaling experiments in Section 3.4. More details about the number of parameters for the various model sizes can be found in Table G.10.

Transformer-XL memory is a cached memory of previous attention layer inputs, concatenated to the keys and values during each forward pass. Inputs to intermediate layer are activations from the previous layer, which in themselves contain information about the past. Caching M activations this way theoretically allows for an effective memory horizon of M x L, where L is the number of attention layers in the network. Therefore, to avoid implicitly scaling effective Transformer-XL memory length, in our model size scaling experiments, we fix the number of layers in the Transformer, and scale parameters only by altering the Transformer embedding size (d_{model}), with the feed-forward size fixed at $4d_{\text{model}}$, as is standard in Transformer architectures (Vaswani et al., 2017).

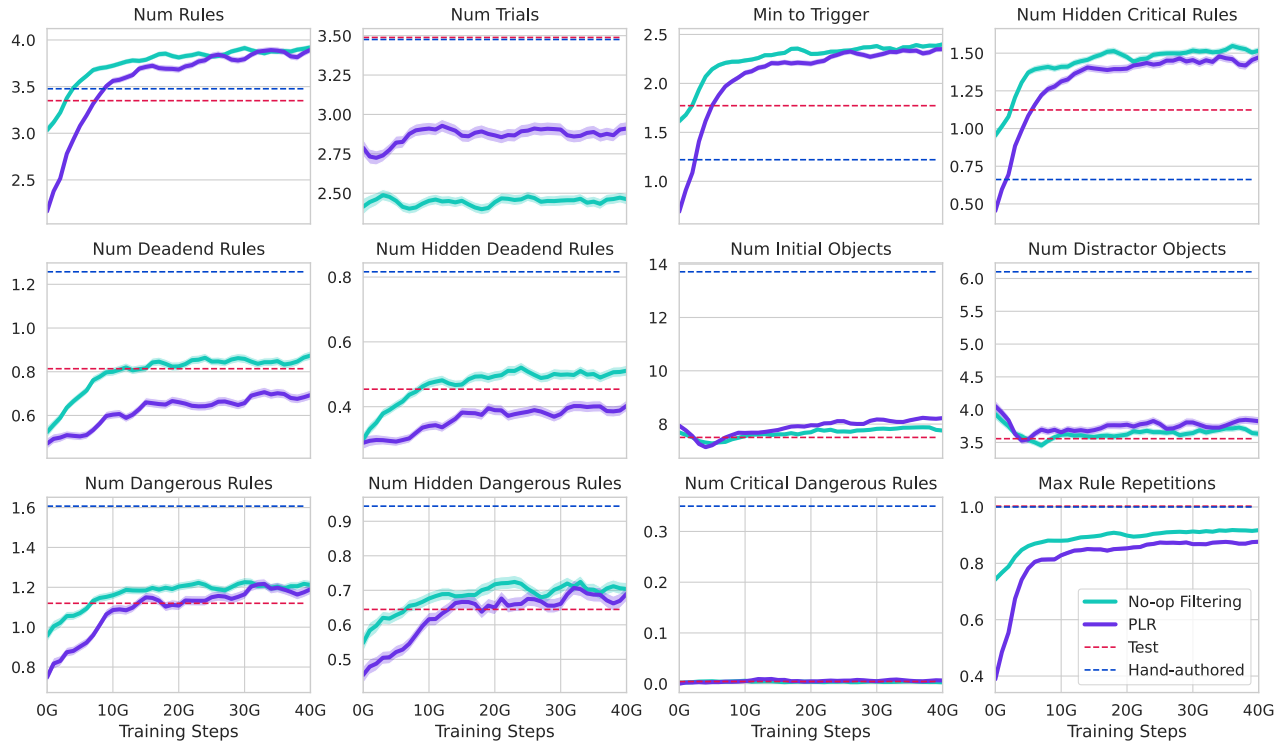


Figure G.3: Emergent curricula for No-op filtering and PLR. Plots show the full set of task metrics for the dynamic training set, averaged over all tasks in the set, with standard error shaded. In all plots, a higher metric value corresponds to greater task difficulty. Horizontal lines show the same metric values averaged over the test (dashed) and hand-authored (dotted) evaluation task sets.

Table G.9: Experimental setup for model size scaling.

Model parameters	Memory	Task pool	Curriculum	Teacher	Steps
6M TXL / 41M total 23M TXL / 76M total 42M TXL / 112M total 57M TXL / 141M total 75M TXL / 175M total 169M TXL / 353M total 265M TXL / 533M total	1800	25B	No-op	Table G.8	75B

Table G.10: Transformer hyperparameters for different model sizes.

Model parameters	Embedding size	Blocks	Key size	Value size	Heads	FFW size
6M TXL / 41M total	288	6	48	48	6	1152
23M TXL / 76M total	576	6	48	48	12	2304
42M TXL / 112M total	768	6	32	32	24	3072
57M TXL / 141M total	896	6	32	32	28	3584
75M TXL / 175M total	1024	6	32	32	32	4096
169M TXL / 353M total	1536	6	48	48	32	6144
265M TXL / 533M total	1920	6	48	48	40	7680

G.9. Scaling the memory length

Table G.11 shows the details of the experimental setup for the memory length scaling experiments in Section 3.4. We show the effective memory timesteps for each experiment, computed as the number of cached network activations times the number of transformer blocks (6).

Table G.11: Experimental setup for scaling the memory length.

Model Parameters	Memory	Training task pool	Curriculum	Teacher	Training steps
23M TXL / 76M total	600 1800 3000 4200	200M	No-op	Table G.8	25B

G.10. Scaling the size of the task pool

Recall that in XLand, a task is the combination of a world (the physical layout of terrain and objects) and a game (specifying the goal and production rules). We investigate the effects of training on tasks sampled from a small pool of 200M distinct tasks (4,000 worlds $\times$ 50,000 games) compared with a large pool of 25B distinct tasks (50,000 worlds $\times$ 500,000 games). Table G.12 shows the details of the experimental setup for scaling the size of the task pool in Section 3.5.

Table G.12: Experimental setup for scaling the task pool size.

Model parameters	Memory	Training task pool	Curriculum	Teacher	Training steps
23M TXL / 76M total	1800	200M 25B	No-op	Table G.8	25B
75M TXL / 175M total		200M 25B			

G.11. Scaling the complexity of the task pool

Table G.13 shows the details of the experimental setup for scaling the complexity of the task pool in Appendix H.3. In this experiment, the distillation teachers are different for the two agents we compare. Therefore we cannot disentangle the effects of distillation and task complexity. Nevertheless, the results remain indicative of the importance of task complexity. The teacher for the task distribution across multiple world topologies is trained as in Table G.8. The teacher for the task distribution in a single room comes from a long lineage (¿ 6 generations) of distillation teachers starting with agents trained on XLand 1.0 (OEL Team et al., 2021).

Table G.13: Experimental setup for scaling the complexity of the task distribution.

Model parameters	Memory	Task pool	Curriculum	Steps
6M TXL / 41M total 23M TXL / 76M total 42M TXL / 112M total 57M TXL / 141M total 75M TXL / 175M total	1800	4k worlds $\times$ 50k games	No-op	23B
6M TXL / 41M total 23M TXL / 76M total 42M TXL / 112M total 57M TXL / 141M total	1800	1 world $\times$ 5k inits $\times$ 50k games	No-op	23B

G.12. Distillation enables scaling agents

Table G.14 shows the experimental setup for the distillation experiments in Section 3.6.

Table G.14: Experimental setup for distillation experiments.

Model Parameters	Memory	Training task pool	Curriculum	Teacher	Steps
TXL 23M TXL / 76M total	1800	See Sec G.4	PLR (G.6)	Table G.3	22B
				None	
TXL 265M TXL / 533M total				Table G.3	
				None	

G.13. Training on more trials with skip memory

Table G.15 shows the details of the experimental setup for the memory scaling experiments in Section H.6. This is the same setup as in Table G.15, except for the number of training steps and the variation of memory architecture and training trials discussed in the main text.

Table G.15: Experimental setup for experiments training on more trials with skip memory.

Model Parameters	Memory	Training task pool	Curriculum	Teacher	Steps
23M TXL / 76M total	1800	200M	No-op	Table G.8	50B
	$1800 \times 4 = 7200$				

H. Additional Experiments

H.1. Multi-agent adaptation

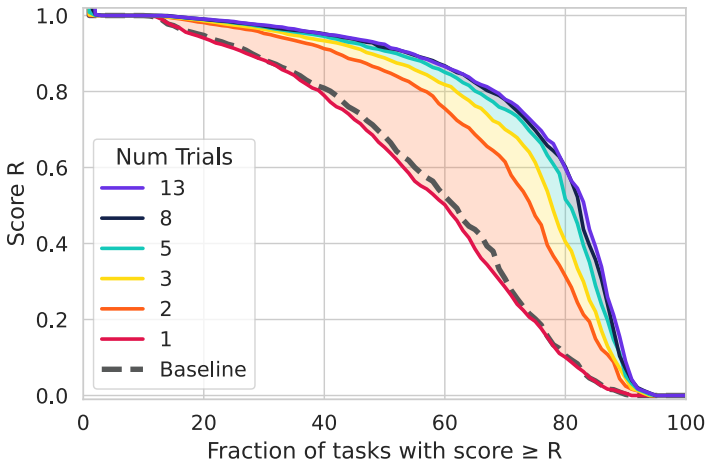


Figure H.1: We report the distribution of normalised task scores over the multi-agent task test set when evaluated with various numbers of trials. All tasks are evaluated in cooperative self-play. On the y-axis is the total last-trial reward relative to that of an agent fine-tuned on the test tasks (approximating "infinite trials" performance). Curves moving further towards the top right corner indicate better performance. When given more trials, the agent achieves higher scores in the last trial, showing test-time adaptation across most of the task distribution (shaded regions).

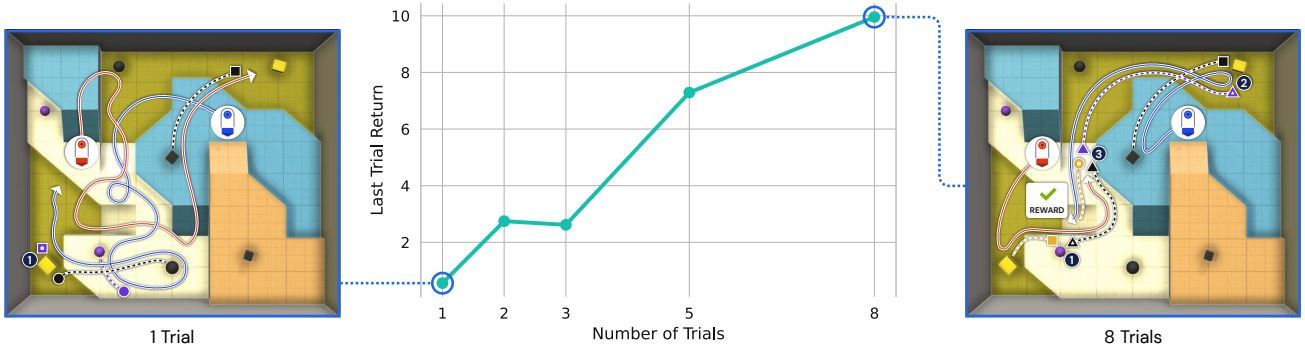


Figure H.2: Average performance and representative behaviour of AdA on the probe task `Irreversible Production for Two` when evaluated in self-play with various numbers of trials. AdA's performance increases when given more trials, showing test-time adaptation. The top-down view images show representative last-trial trajectories when given different numbers of total trials.

Figure H.1 demonstrates adaptation across a wide range of percentiles on a test-task set of multi-agent tasks. Figure H.2 demonstrates last-trial performance of AdA in one particular probe task. To generate these plots, AdA was trained as described in Section 3.1 (Multi-agent).

H.2. Conditioning on number of shots doesn't affect agents' performance

Figure H.3 shows the score obtained by AdA for each percentile, in trial 1 of episodes with only 1 trial ($k = 1$) and in trial 1 of episodes with 8 trials ($k = 8$) in our held-out test set. The overlap of these lines indicates that AdA does not use the trial conditioning information it observes to adjust its behaviour in any way that affects its score. If the agent were to follow a more exploratory policy when it has more trials, we might expect the scores of trial 1 with $k = 8$ to be lower than the score of trial 1 with $k = 1$.

This may be the optimal policy for our XLand 2.0 tasks, or it may reveal a limitation of our training procedure. One

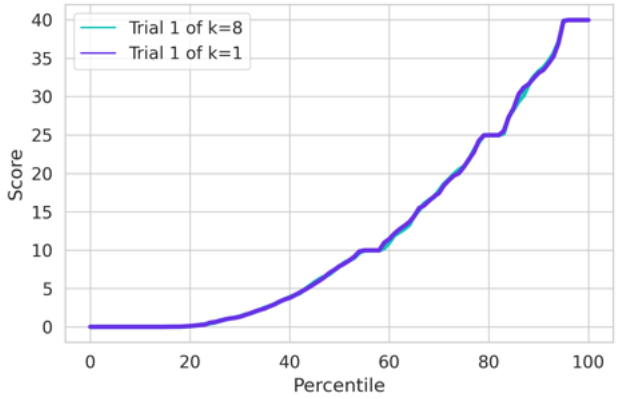
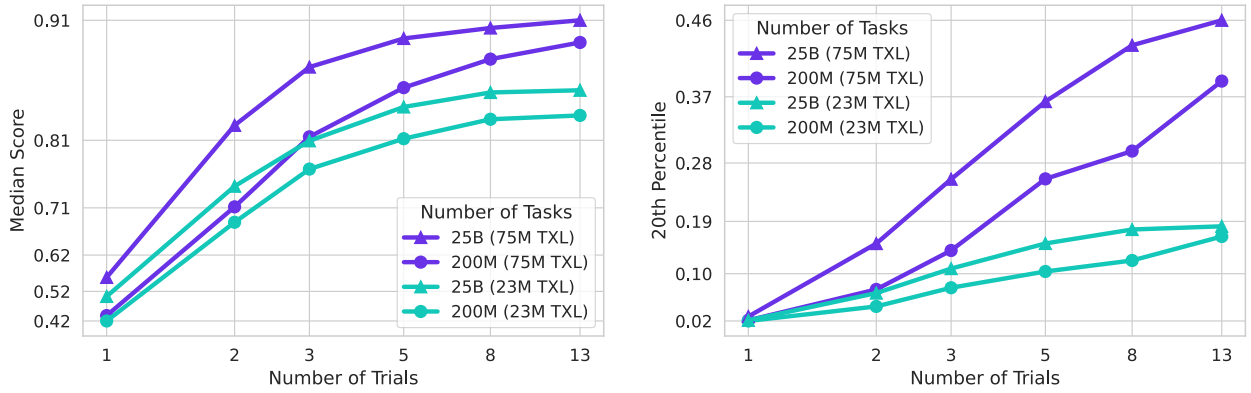


Figure H.3: A comparison of the first-trial score in episodes with 1 trial and episodes with 8 trials. The lines are almost perfectly overlapping, which indicates that our agent does not leverage number-of-trials conditioning information to adjust its policy to a more exploratory one in early trials when more trials are available.

can imagine a scenario in which, knowing that there are 8 trials in total, a Bayes-optimal policy chooses to display a less rewarding and more exploratory behaviour in trial 1, compared to how it would behave if told that there was only a single trial in which to collect reward. For instance, an agent may be able to guarantee a deterministic reward later, having discovered some key information, at the cost of foregoing an stochastic reward early on. We did not directly incentivise this behaviour in our training process. In fact, we may have discouraged it, since AdA learns from all rewards in an episode (not just in the last trial), and with a discount factor smaller than 1, which could lead to myopic behaviour.

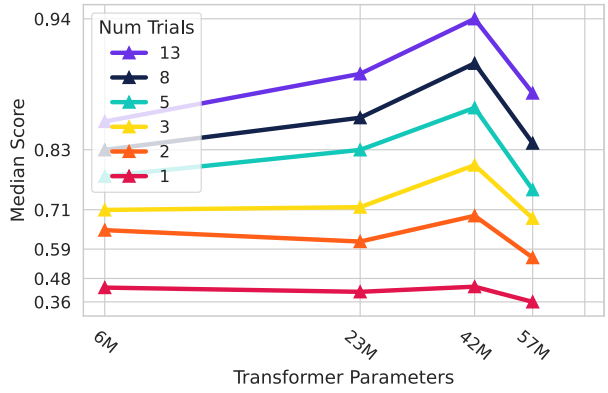
H.3. Scaling complexity of the task pool



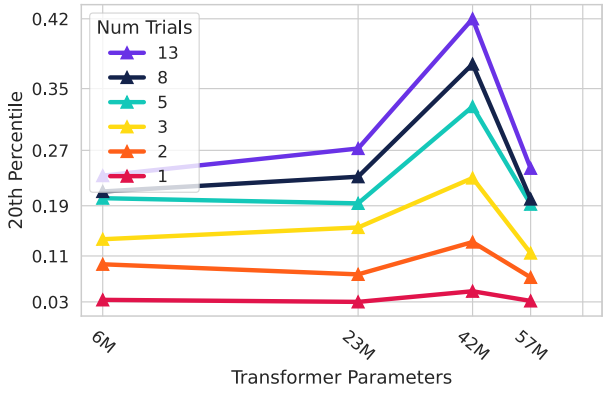
(a) (b)

Figure H.4: Median **(a)** and 20$^{\text{th}}$ percentile **(b)** adaptation scales with the size of the task pool. The effect is especially prominent for larger models. We show the y-axis on a logarithmic scale as in the other scaling experiments. Here, we plot number of trials on the x-axis and examine the gaps between the curves for the two task distributions (triangle markers vs. circular markers).

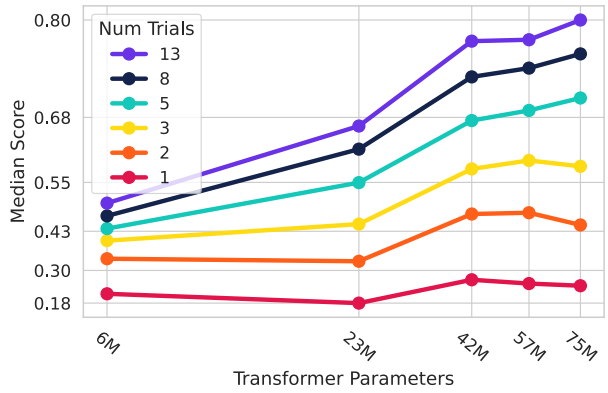
One final axis along which it is possible to scale our method is the overall complexity of the task distribution. We compare our main task distribution, as described in Section 2.1 and Appendix D.2 to a subset which maintains the use of the same goals, production rules, and objects, but eliminates any navigational complexity by having a single world topology: an empty room. Recall that we count the number of tasks as the product of number of worlds and the number of games. To disentangle the effects of scaling complexity versus scaling the sheer number of tasks, we add 5,000 unique object initialisation points for the empty room. These serve as the proxy "4,000 worlds" and are, by design, much less diverse



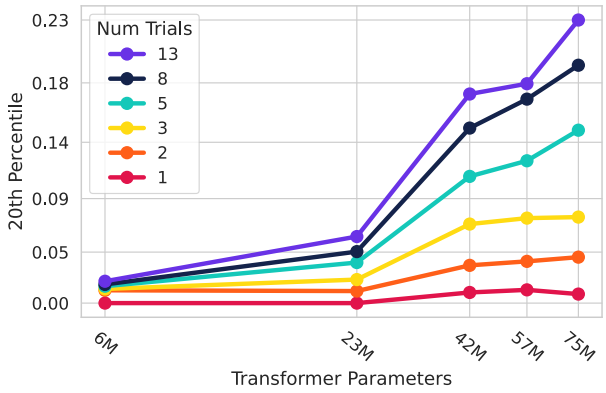
(a) Task distribution with only empty room inhibits scaling (median).



(b) Task distribution with only empty room inhibits scaling (20th percentile).



(c) Task distribution with many world topologies facilitates scaling (median).



(d) Task distribution with many world topologies facilitates scaling (20th percentile).

Figure H.5: The benefit of scaling model size is bottlenecked if the distribution is not complex enough, even if the total number of tasks is accounted for.

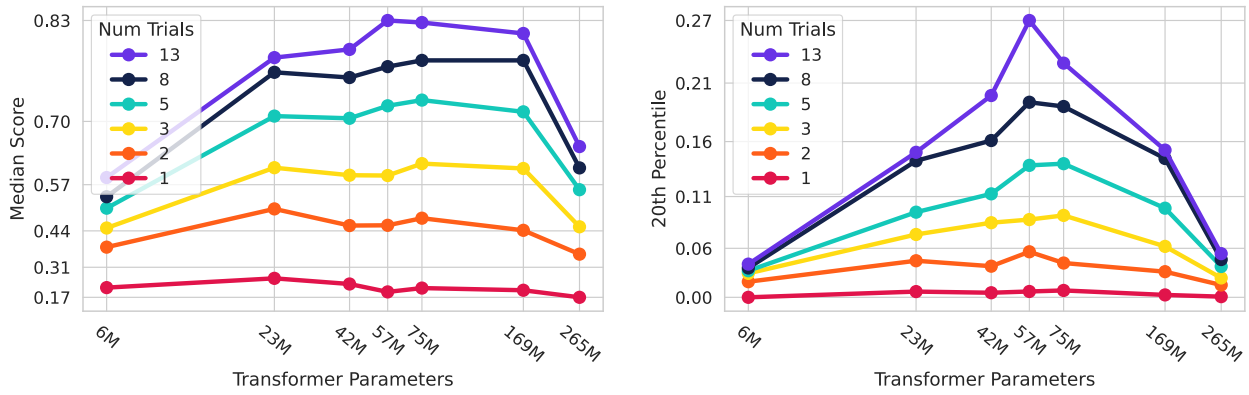
and complex than the 4,000 worlds in the main training pool.[3] For more details of the experimental setup, see Appendix G.11 and Table G.13.

In Figure H.5, we show that low environment complexity can be a bottleneck to scaling, by comparing the effectiveness of model scaling between agents trained on the two distributions, and each evaluated on their respective test sets. On both the median (Figure H.5a) and 20th (Figure H.5b) percentiles in the empty room, we see that past a certain point (42M Transformer parameters), scaling model size begins to reduce performance. By contrast, in the distribution with many world topologies (Figures H.5c and H.5d), increased model size continues to improve performance far beyond this, showing improvements through at least 75M Transformer parameters. Open-ended settings with unbounded environment complexity, such as multi-agent systems, may therefore be particularly important for scaling up adaptive agents.

H.4. Computational cost

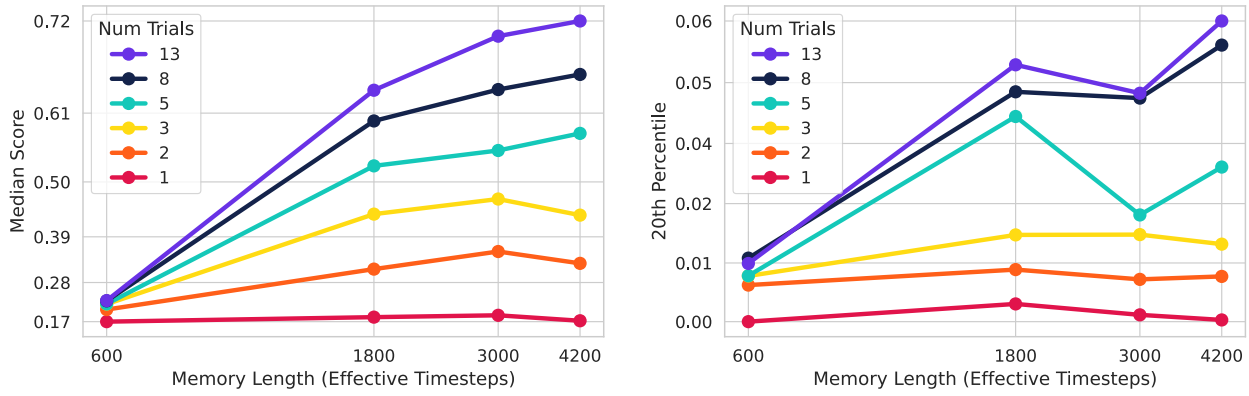
In the scaling experiments (Sections 3.4 and 3.5), we compare agents after they have been trained for an equivalent number of steps. While this controls for sample efficiency of models, here we provide an analysis of the computational cost in FLOPs for a given experiment, and reproduce some of our scaling results, controlling for for compute cost. We see that bigger is not always better from this perspective. For each model size and memory length we use JAX (Bradbury et al.,

[3]Note that the distribution over world topologies we use here is smaller than the distribution used in the model scaling experiments in Section 3.4, and results are therefore not comparable across these sections.



(a) (b)

Figure H.6: Scaling Transformer model size controlling for the total number of FLOPs for the learner and actors, including auto-curriculum evaluation actors.



(a) (b)

Figure H.7: Scaling Transformer-XL memory controlling for the total number of FLOPs for the learner and actors, including auto-curriculum evaluation actors.

Table H.1: FLOPs per frame for different model sizes and memory lengths.

Model parameters	Memory	Learner FLOPs per frame	Actor FLOPs per frame
6M TXL / 41M total		1,138,005,632	736,987,392
23M TXL / 76M total		1,380,255,078	2,580,445,440
42M TXL / 112M total		1,623,461,663	4,489,239,552
57M TXL / 141M total	1800	1,821,422,230	6,063,742,976
75M TXL / 175M total		2,048,253,663	7,879,863,808
169M TXL / 353M total		3,243,189,525	17,560,326,144
265M TXL / 533M total		4,443,008,234	27,358,181,376
	600	1,278,491,404	963,739,136
23M TXL / 76M total	3000	1,481,009,258	4,181,042,688
	4200	1,582,270,213	5,789,702,144

2018) cost analysis to estimate the number of FLOPs per frame of the learner step and actor step (Table H.1).

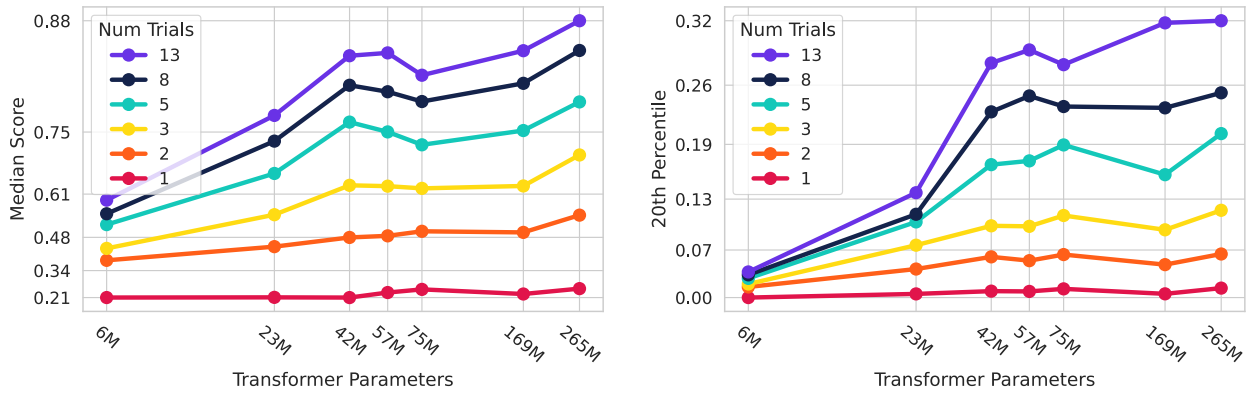
We multiply the values in Table H.1 by the number of learner/actor steps for each experiment, then, for a given comparison, we take the largest such value common to all experiments (usually associated with the smallest model) as the total FLOPs,

and make the comparison of each model at this number of FLOPs. The results for the FLOPs-matched model scaling experiments are shown in Figure H.6. We see a reduction in performance as the model size grows beyond a "sweet spot" around 57M Transformer parameters (141M total parameters). Results for the FLOPs-matched memory scaling experiments in Figure H.7 show that there is still benefit to increasing context lengths given a fixed computational budget. Details of the compute used for these experiments can be found in Tables H.2 (model size) and H.3 (memory).

We note that Table H.1 indicates poor scaling of actor step FLOPs with model size, and suspect this could be due to poor optimisation of a single-step query of the Transformer on TPU, compared to the operation batching achieved with a rollout length of 80 on the learner. In order to account for this and for potential discrepancies in number of actor steps due to differences in curriculum evaluation based on model quality, we also provide plots which only account for the learner step FLOPs for each model: Figures H.8 (model size), and H.9 (memory length). These might be more informative, and show that performance still increases as a function of model size and memory length, albeit not as steeply as when controlling for sample efficiency directly. Details of the compute used for these experiments is shown in Tables H.4 (model size) and H.5 (memory length).

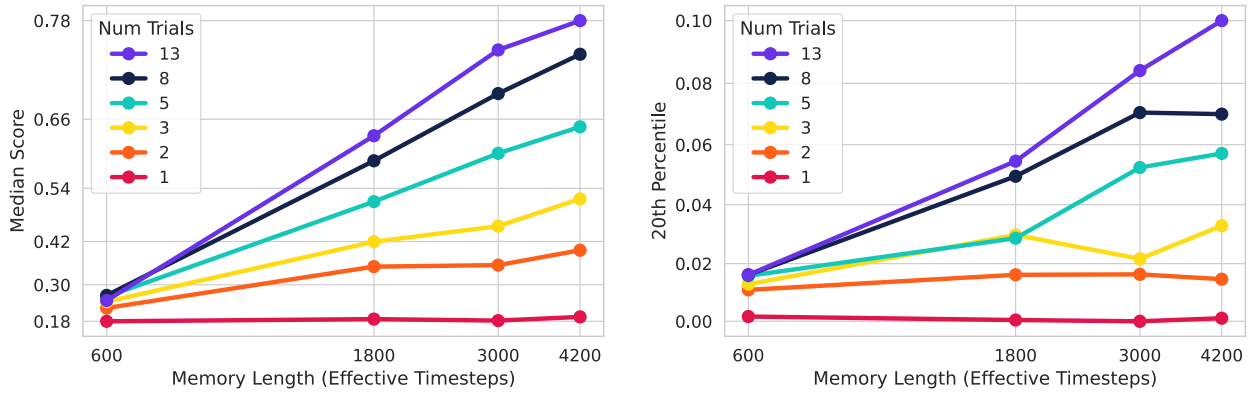
H.5. Repeated distillation

In Section 3.6, we show that distilling an agent into an identical student can lead to large increases in the agent's performance. Here, we investigate the potential benefits of applying the procedure repeatedly. To this end, we continue the experiment shown in Figure 11 and add a third generation, using a snapshot taken from the previous student after 25 billion



(a) (b)

Figure H.8: Scaling Transformer-XL model size controlling for the number of learner FLOPs.



(a) (b)

Figure H.9: Scaling Transformer-XL memory length controlling for the number of learner FLOPs.

Table H.2: Corresponding learner steps given total FLOPs for each model size.

Model parameters	Memory	Total FLOPs	Learner steps
6M TXL / 41M total	1800	2.0×10^{20}	97B
23M TXL / 76M total			44B
42M TXL / 112M total			27B
57M TXL / 141M total			23B
75M TXL / 175M total			18B
169M TXL / 353M total			9B
265M TXL / 533M total			5B

Table H.3: Corresponding learner steps given total FLOPs for each memory length.

Model parameters	Memory	Total FLOPs	Learner steps
23M TXL / 76M total	600	7.9×10^{19}	31B
	1800		17B
	3000		11B
	4200		8B

Table H.4: Corresponding learner steps given learner-step-only FLOPs for each model size.

Model parameters	Memory	Learner step FLOPs	Learner steps
6M TXL / 41M total	1800	1.0×10^{20}	92B
23M TXL / 76M total			76B
42M TXL / 112M total			65B
57M TXL / 141M total			58B
75M TXL / 175M total			51B
169M TXL / 353M total			32B
265M TXL / 533M total			23B

Table H.5: Corresponding learner steps given learner-step-only FLOPs for each memory length.

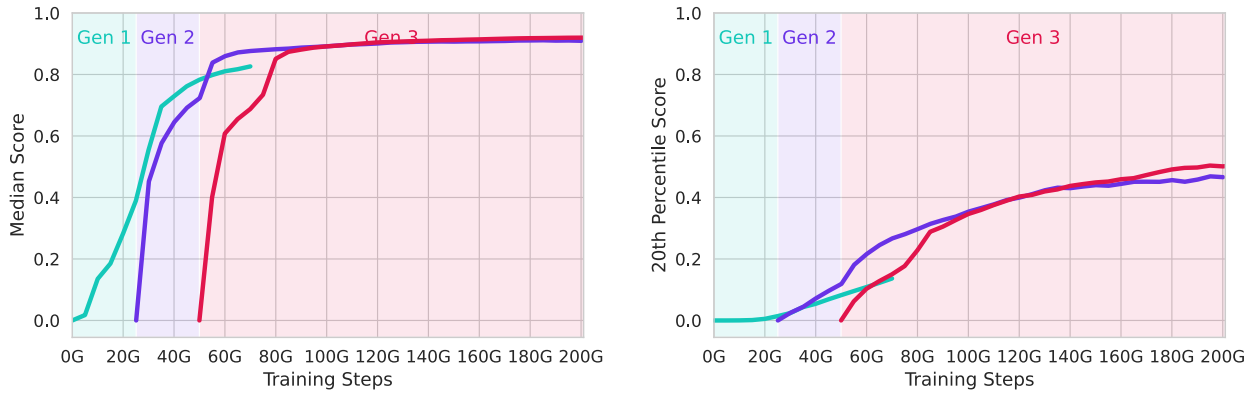
Model parameters	Memory	Learner step FLOPs	Learner steps
23M TXL / 76M total	600	3.9×10^{19}	39B
	1800		36B
	3000		29B
	4200		25B

frames, equivalent to 50 billion frames of total experience when taking the teacher's experience into account. Figure H.10 shows that applying this procedure repeatedly can indeed lead to additional benefits; however, we observe diminishing returns with successive generations.

H.6. Training on more trials with skip memory enables many-shot adaptation

So far, we have considered the few-shot regime in which we train on 1 to 6 trials and evaluate up to 13 trials. In this section, we evaluate AdA's ability to adapt over longer time horizons. We find that when trained with $k \in \{1, 2, \ldots 6\}$, agents do not continue to adapt past 13 trials; however, this long-term adaptation capability is greatly improved by increasing the maximum number of trials during training to 24 and increasing the effective length of the memory accordingly. These results show that our method naturally extends to many-shot timescales, with episodes lasting in excess of 30 minutes.[4] In this section, we ablate both factors separately, and show that both are important for long-range adaptation. The training configuration (which is identical to that of the memory scaling experiments) is detailed in Table G.15.

[4] 48 trials of a 40s task lasts for 32 minutes. By contrast, the average length of a Starcraft 2 game is between 10 and 15 minutes, and AlphaStar acted less frequently per-second than AdA does (Vinyals et al., 2019).



(a) (b)

Figure H.10: Normalised few-shot score over three generations using the 23M parameter Transformer-XL. The first and second generations correspond to the agents shown in Figure **??**. The third generation is distilled from the second after it has been trained for 25 billion steps. The x-axis counts the combined amount of experience, starting from the experience collected by the original teacher. The third generation shows additional gains over the second, but to a lesser degree than the gap between the first and second generations.

As we noted in Section 3.4, increasing the memory length leads to increased capacity that benefits the agent even when the entire episode fits in memory, but also comes at the cost of increased computation. To disentangle these factors, we propose a simple change to the memory architecture described in Section 2.4 which increases effective memory length without increasing computational cost. We use a GRU to encode trajectories before feeding them to the Transformer-XL. This allows us to sub-sample timesteps from the encoded trajectories, enabling the agent to attend over 4 times as many trials without additional computation. We show that the additional GRU on its own does not affect the performance of the agent greatly.

As can be seen in Figure H.11a, increasing the number of trials in the training distribution significantly boosts performance in later trials, especially when the memory length is scaled accordingly. In other words, the adaptation strategy learned by AdA benefits from experiencing a large number of trials, rather than just very recent ones. Therefore we can conclude that AdA is capable of adaptation based on long-term knowledge integrated into memory across many trials, as opposed to merely encoding a simple meta-strategy that only depends on the trajectory from the previous trial.

Increasing the number of trials in training leads to better adaptation even in the absence of increased memory. This indicates that the agent is able to learn better exploration and refinement strategies when afforded longer training episodes consisting of more trials. Note that increasing effective memory without increasing the number of training trials does not improve performance, as the agent has not been trained to make use of the additional memory capacity.

I. Human-Timescale Adaptation

In this section we provide more details regarding our claim of human-timescale adaptation.

I.1. Probe tasks

Tables I.1 and I.2 describe the single-agent and multi-agent probe tasks respectively. Unless noted otherwise, all probe tasks are set in complex worlds with many objects, and use multiple production rules fully hidden from the players.

Table I.1: Single-agent probe tasks

Name	Description

Wrong pair disappears	The player's goal is to hold a black cube, which does not exist among the initial objects. There are two (hidden) rules. The player needs to identify the correct world state which triggers the rule that creates the cube and not the one which destroys the necessary inputs. All this is embedded in a challenging world layout with one-way drops and limited visibility.
Wrong pair disappears, partial hiding	'Wrong pair disappears', but instead of hiding all rules completely we only hide the input objects for both rules (outputs and conditions are fully visible).
Irreversible production	Similar to 'Wrong pair disappears', but this task has multiple dead ends (rules which create unsolvable states).
Irreversible production, all rules visible	'Irreversible production', but with all rules fully visible to the player.
Push, don't lift	The vast majority of training and evaluation tasks require lifting objects. Here two hidden rules destroy any object when lifted. In order to create the goal state, some "lateral thinking" is necessary: the player needs to identify that pushing the cubes with their body is possible.
Push, don't lift, with distractors	Similar to 'Push, don't lift', but here a large number of distractor objects in the world make this a much more challenging exploration task for any player ignoring the objects mentioned in the goal.
Spacer tool	Two objects need to be brought close together, but lifting them or touching them (with the avatar's body) destroy them. The solution is to use another object in the world as a tool to push them together.
Transform to transport	Again, two objects need to be brought close to each other, but lifting and touching them destroys them. The solution here is to exploit a set of rules that can turn one of the objects into something that can be carried safely, and then turn it back into the original object once it is in the right place.
Small workstation	A very hard object handling task. 8 objects near the player's spawn position need to be combined in different ways and 5 rules need to be triggered (some multiple times) to create the goal object. This task is set on top of a tall plateau and it is very easy to fail by touching an object too hard and throwing it off the edge of the plateau.
Small workstation, all rules visible	The same as 'Small workstation', but all rules are visible to the player.
Crafting pyramid	Eight objects need to be recursively combined first into four, then two and then ultimately one final object. This requires triggering a chain of 7 rules. This seems easy for humans. But the lack of intermediate reward makes this a hard hierarchical credit assignment task for agents.
Crafting tree, all rules visible	Similar to the crafting trees in video games like Minecraft, this multi-step task requires triggering different rules in a chain to create the goal object. All objects exist in the world multiple times, making many different solutions viable.
Crafting tree, hidden shortcut	Identical to 'Crafting tree, all rules visible', but one additional (hidden) rule exists. This allows the player to take a shortcut that lets them finish the task faster than executing only the visible rules.
Antimatter	In a world full of yellow and black spheres, the goal is for no pair of black and yellow spheres to "see each other" (no direct line of sight). Beyond moving the objects and blocking line of sight with the avatar there exists a production rule which destroys any yellow sphere and black sphere pair which touch (similar in spirit to matter and antimatter particles). This is all embedded in a world requiring advanced navigation skills.

Antimatter with creation	Similar to 'Antimatter', only here a third object (purple pyramids) exists which duplicates any sphere touching it. In addition, this task is set on two plateaus, making it easier to break line of sight and reducing the navigation challenge.
Pyramid in a haystack	To create the necessary yellow pyramid, the player needs to find and hold the purple cube. There are several distractor objects and distractor rules in this world, requiring the player to deal not just with a hard exploration challenge but also a very noisy environment.
Protect the egg	The player is tasked to hold the single yellow sphere in the world. A large number of other spheres exist in the world. These destroy the yellow sphere on collision. As these touch each other or get near the player they get duplicated. This can lead to a constantly growing number of objects, filling up the world.
3 spheres jumbled	3 spheres exist in the world. Holding one of them creates the goal object. Only one sphere can be reached within the 10-second time limit, meaning that the optimal policy on the first trial is to choose uniformly at random.
Two doors	The goal object is hidden behind one of the two large objects (the "doors") positioned at opposite ends of the world. Only one of them can be reached in time, so the player needs to decide between exploring one of them per trial.
Same signature: match colours	All 'Same signature' tasks have the exact same world layout, goal and number of fully-hidden rules. This means they look exactly the same to a player starting out. This one only requires two objects of matching colour to be brought close together to create the goal object, using only one production rule out of three.
Same signature: three steps	This 'Same signature' task variant requires the player to trigger all three (hidden) production rules to create the goal object.
Same signature: two dead ends	In this 'Same signature' task variant, two of the tree rules are dead ends, leading to an unsolvable world state. Only one rule is helpful (and in fact required) to solve the task.
Same signature: destroy to protect	To solve this 'Same signature' task variant the player first needs to destroy a black sphere (by getting near it) before creating the goal object (a yellow cube). Otherwise when the black sphere "sees" the yellow cube, both get destroyed. This is a very hard credit assignment challenge even for human players (it is hard to notice what is going on).
Don't act	This task is pre-solved: the player is getting reward from the very beginning. If they lift or touch one of the two objects in the world, the object gets destroyed and reward is now impossible.
Don't peek	This task is 'pre-solved': the player is getting reward from the very beginning. However, the player will destroy any object they look at, at which point reward is impossible. So the optimal strategy is to not look at anything but the sky.
Navigation: find the cube	This memory task is not using production rules. It uses a large world with the goal object (a cube) hidden after a very winding path.
Navigation: find the cube with teaser	This memory task is not using production rules. It is set in a large world with the goal object (a cube) hidden after a very winding path. The object is visible from the spawn point but out of sight after starting to traverse the terrain.
Navigation: hold up high	This memory task is not using production rules. The goal object (a pyramid) is hidden on top of a plateau and can only be seen when nearly there.

Object permanence: yellow cube	This memory task is not using production rules. The goal object (a yellow cube) is visible from the spawn point. After moving for a bit, a decision between two paths has to be made, with the correct path being to the right. At this point the cube is no longer visible.
Object permanence: black cube	This memory task is not using production rules. This has same world layout as the task above, only here the player is asked to find the black cube, which requires going to the left.

Table I.2: Multi-agent probe tasks

Name	Description
Pass over the wall	The players are separated by an opaque wall. They cannot see each other, only their half of the world. The solution requires "passing" the accessible objects on either player's side to other player to combine them into the goal object. Then the players must make sure the correct player holds this object.
Pass over the wall repeatedly	Similar to 'Pass over the wall' but here 2 out of 3 initial objects are 'frozen' (cannot be moved). This prescribes a very specific solution strategy that requires the players to pass 3 objects over the wall in a specific order.
Coordinated production	This task requires each player to be near a sphere to turn this sphere into a pyramid, and then for both pyramids to be touching each other to create the goal object. The spheres are slightly hidden in a complex world, requiring some exploration.
Coordinated production with deadends	Like 'Coordinated production' but here each player will destroy one of the initial objects if they get near it. These dead-end rules make this a much harder exploration problem.
Coordinated exchange	This requires each player to create a new object by holding an existing object, then to hold the object created by the other player to turn it into another intermediate object and finally for both objects to be combined into the goal object. While the world is fully accessible to both players, this can only be solved if both players actively participate.
Overcooked: coordination ring	Inspired by the video game Overcooked (Carroll et al., 2019; Strouse et al., 2021), in this task both players need to "serve tomato soup to a hungry patron". This is implemented as a repeatable four-step production rule chain which requires both players to traverse their shared space (a circular layout) carefully in order to not block the other player.
Overcooked: coordination ring, all rules visible	While in 'Overcooked: Coordination ring' all production rules are hidden from both players, here they are fully visible (to match the dynamics of the original Overcooked game).
Overcooked: cramped room	Similar to 'Overcooked: coordination ring' but with a different layout for the shared space and a different number of initial objects, using different shapes and colours.
Overcooked: cramped room, all rules visible	While in 'Overcooked: cramped room' all production rules are hidden from both players, here they are fully visible (to match the dynamics of the original Overcooked game).
Overcooked: forced coordination	Similar to the other 'Overcooked' task variants, but here both players are restricted to only a certain part of the world and so are forced to coordinate to solve this task. No player can solve this alone since they cannot reach all initial objects.

Overcooked: forced coordination, all rules visible	While in 'Overcooked: forced coordination' all production rules are hidden from both players, here they are fully visible (to match the dynamics of the original Overcooked game).
Kickball	This task is set in large world with two frozen pyramids on opposite sides of the world. Both players want to bring all of the plentiful purple spheres to the yellow pyramid. But lifting them destroys the pyramids so they need to "kick" them by bouncing them off the avatar. Think "soccer practice".
Lemon eater	The first player destroys all yellow spheres (of which there are many) when bumping into them. This is also the goal for both players. So they need to cooperate to bring all yellow spheres to the first player as quickly as possible.
Careful lemon eater	Like 'Lemon eater' but any collision between two spheres turns them from yellow to purple. Purple spheres are "not edible" and so the players need to be careful to not create those, otherwise they will lose out on reward.
Lemon eater and maker	Like 'Lemon eater' but we start out with only purple spheres. Only the second player can turn purple into yellow spheres by lifting them, effectively having to "create food" for the first player.
Antimatter for two	Identical in nature to the single-player 'Antimatter' task but set in a different world layout and with two players who share the same goal.
Antimatter with creation for two	Identical in nature to the single-player 'Antimatter with creation' task but set in a different world layout and with two players who share the same goal.
Antimatter with copies for two	Identical to 'Antimatter for two' but here any two spheres of the same colour colliding leads to the creation of another sphere of that colour. This can set in motion runaway growth in the number of spheres, making it very hard to solve the task.
Irreversible production for two	Identical in nature to the single-player 'Irreversible production' task but set in a different world layout and with two players who share the same goal.
Irreversible production for two, all rules visible	Like 'Irreversible production for two' but with all production rules visible to both players.
Wrong pair disappears for two	Identical in nature to the single-player 'Wrong pair disappears' task but set in a different world layout and with two players who share the same goal.
Wrong pair disappears for two, partial hiding	Like 'Wrong pair disappears for two' but with only the input objects of the productions rules being hidden, the outputs and condition being visible.
Crafting pyramid for two	Identical in nature to the single-player 'Crafting pyramid' task but set in a different world layout and with two players who share the same goal.
Information asymmetry	A simple world in which the players are asked to execute a two-step production rule in the presence of multiple dead ends. While the first player knows all the rules, they are completely hidden from the second player. This task is intended to measure a specific flavor of third-person imitation.
Information asymmetry with repetition	While 'Information asymmetry' only allows for up to four completions, this task variant is (given unlimited time) infinitely repeatable, providing more opportunities for imitation.
Combine outputs	Similar in nature to 'Coordinated production' but set in a larger world and (through the use of frozen objects) requiring both players to repeatedly navigate quite far to create input objects for shared creation.

Two machines	Many objects of different colours and shapes litter this world. The frozen yellow pyramid on one end of the world transforms these objects into an intermediary object. The players then need to bring the intermediary object to the frozen black pyramid on the other end of the world to "cash it in" for instantaneous reward, at which point the intermediary object is destroyed. Therefore to get more reward, the players must repeat this process.
Two machines with information asymmetry	Like 'Two machines', but all rules are visible to one player, and all rules are hidden from the other player. This creates an information asymmetry and thus an opportunity for third-person imitation.

I.2. Comparing human and agent scores on every probe task

In Figure I.1 we show the raw last-trial total reward for humans and our agent as a function of number of trials across every one of the 30 evaluation probe tasks.

I.3. Quantifying stochasticity

Task variation. Figure I.2a shows that there is fairly high variance in the score obtained by a single agent over 50 repetitions of a single task. This is due to random spawn rotations of the avatar following every environment reset and stochasticity in the agent's policy. To address this we run 50 repetitions of every (task, k) combination and average the score over these repetitions. This reduces the standard deviation to that shown in Figure I.2b. Here, the standard deviation of the mean task score reduced from a maximum of 0.43 with 1 repetition, to a maximum of 0.06 with 50 repetitions.

Agent initialisation variation. After accounting for task variation, Figure I.3 shows the variance due to agent initialisation seed during training. We plot the score as a function of k on our test set for the 76M total parameter version of AdA with 5 different initialisation seeds. The maximum standard deviation observed for any number of trials in the median was 0.04 and for the 20$^{\text{th}}$ percentile was 0.02. This low initialisation seed variance led us to run our ablations with one initialisation seed to save on compute. We note that the results shown in our ablation section have significantly larger than one standard deviation differences.

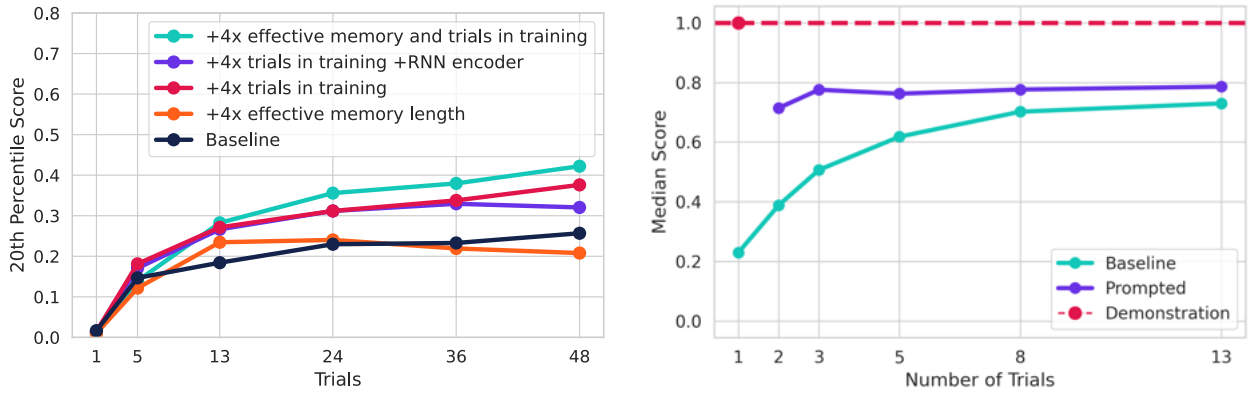
I.4. Prompting through first-person demonstrations

Figure I.4 shows the performance of AdA prompted with a fine-tuned agent compared to an unprompted baseline on each of the 30 single agent hand-authored probe tasks. The figure reveals a set of tasks on which AdA is able to leverage information in the prompt, resulting in perfect or near perfect scores. There are also tasks where AdA does not seem to be able to do this. In all but one case, prompting does not hurt performance. Two videos in the supplementary files compare the behaviour when prompted and when not prompted on the task `Object permanence: yellow cube`.

Analysing the tasks in Figure I.4 suggests that prompting is useful for short navigational tasks such as `Navigation: hold up high` in which the agent follows a short and simple path to reach the goal object. Prompting does not, however, improve performance for longer and more complex navigation tasks like `Navigation: find the cube`, likely due to the full demonstration being too long to fit in the agent's memory context.

We observe a similar pattern in tasks involving production rules. For tasks with up to 2 production rules in the solution path, such as `Same signature: match colors`, we observe the unprompted agent exploring different objects to determine the correct rule to trigger. When prompted with a demonstration it subsequently triggers the correct rule immediately and achieves a perfect score. An exception to this is `Same signature: destroy to protect` where one of the production rules involves destroying an object, which the agent does not appear to remember from the demonstration. For tasks using 3 or more production rules like `Same signature: three steps` (3 production rules in the solution path) and `Small workstation` (5 production rules), the agent tends to only remember a subset of the rules to trigger and continues engaging in exploratory behaviour following the demonstration. The performance on these tasks tend to match the unprompted baseline.

Another factor appearing to influence the effectiveness of prompting is the topology and configuration of objects in the world, as seen in the `Small workstation` tasks. While the teacher demonstrations for these tasks present a clean



(a) (b)

Figure H.11: **(a)** Ablation showing the 20$^{\text{th}}$ percentile of test scores as we vary the maximum number of training trials (from a $k = 6$ baseline to $k = 24$) and increase the effective memory size via sub-sampling (from 1800 steps to 7200 steps). Together, these factors enable the agent to adapt over a larger number of trials (lasting over 30 minutes). Increasing the number of training trials has the biggest effect and is a prerequisite for sub-sampling to be effective. This figure furthermore shows that adding an RNN encoder to facilitate sub-sampling does not by itself greatly affect performance. **(b)** Median hand-authored task score of AdA prompted with a first-person demonstration in the first trial of each episode, compared with an unprompted baseline. The prompted score lies strictly above the baseline which indicates that AdA is able to use information from a demonstration prompt to improve its performance. However, the score lies below that of the demonstration which suggests that it is not able to make perfect use of the demonstration.

trajectory, the agent subsequently knocks into and displaces distractor objects, leading to environment states not observed during the demonstration (and thus not recallable from memory). On the other hand, a favourable configuration of objects appears to make the demonstration easier to learn from, as observed in `Irreversible production`. Here the objects required to trigger the last production rule are positioned close together. The agent shows optimal behaviour here despite the task requiring 3 production rules on the solution path. The agent also appears unable to infer from prompting certain more subtle requirements like relative positioning of objects or tool use. This is observed in tasks like `Antimatter` in which prompted AdA is able to trigger the destruction rule but we did not observe it immediately hiding objects from each other.

Prompting may also help eliminate biases that the agent may have acquired during training. This is reflected in the lower score obtained by the unprompted agent for small k in `Object permanence: yellow cube` compared to `Object permanence: black cube`. These tasks are identical except for the goal being to navigate to a yellow cube on the right, or a black cube on the left respectively. This suggests that the agent may have acquired a bias during training to either prefer black cubes or to navigate towards objects on its left. The significantly higher prompted scores on `Object permanence: yellow cube` for small k suggest that a prompt may help the agent overcome these biases.

Prompting with human demonstrations. We prompted AdA with expert human demonstrations in a small selection of 6 hand-authored tasks, depicted in Figure I.5. These tasks were chosen to be a mixture of tasks where AdA excelled with fine-tuned teacher prompting, where it failed with fine-tuned teacher prompting and where even the fine-tuned teacher failed.

The results show the same pattern as those obtained when prompting with a fine-tuned teacher. In both `Navigation: hold up high` and `Object permanence: yellow cube`, prompted AdA achieved close to optimal performance, exceeding both the baseline and the human demonstration. Figure I.6 depicts the latter behaviour in detail. AdA continues to fail to learn from a demonstration in `Navigation: find the cube with teaser` and in both `Spacer tool` and `Transform to transport`, which our fine-tuned teacher also failed at. A successful human demonstration did not unlock any capabilities AdA was previously not capable of demonstrating, suggesting that these tasks are perhaps too far out-of-distribution with respect to AdA's training tasks. The fact that prompting with off-policy human demonstrations is partially successful is worthy of note, and opens an exciting area of future research.

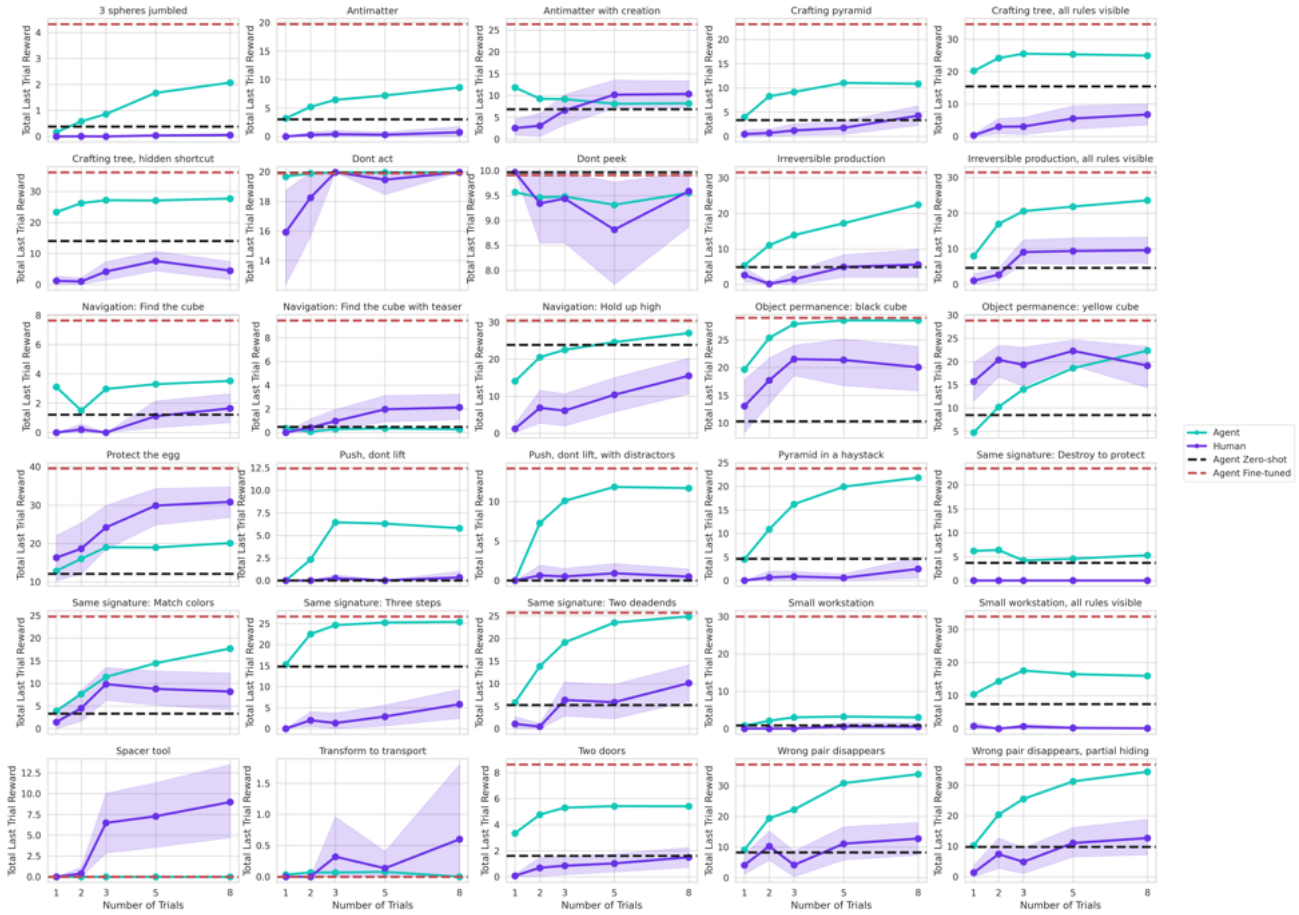
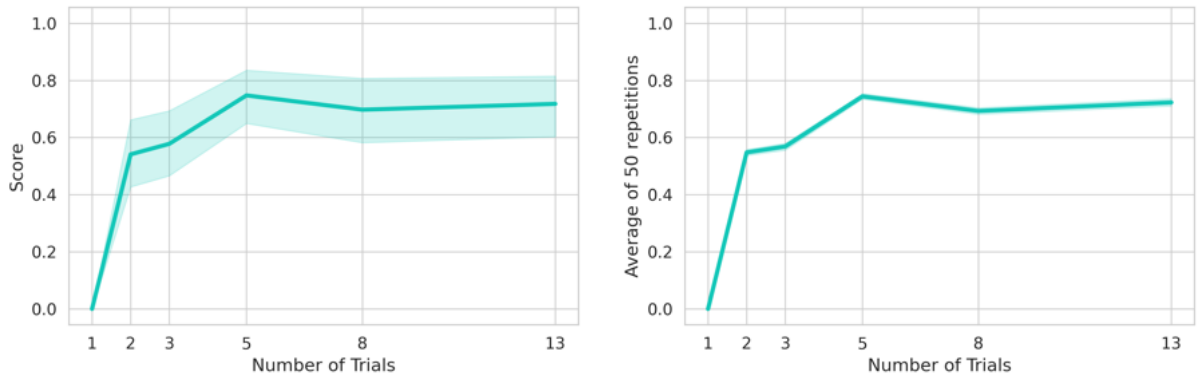


Figure I.1: Comparison of AdA against 19 human players on each of the 30 held-out single-agent hand-authored tasks. The performance of a baseline agent trained to optimise zero-shot performance is shown as a black dashed line and indicates that AdA does not sacrifice its zero-shot generalisation to achieve adaptation. The reward of an agent fine-tuned on the hand-authored tasks is also shown as a red dashed line to provide some indication of the maximum reward achievable in a trial of each task.



(a) (b)

Figure I.2: **(a)** Mean single-repetition score and 95% confidence intervals over 50 samples. **(b)** Mean of the aggregated 50-repetition score and 95% confidence intervals over 50 samples.

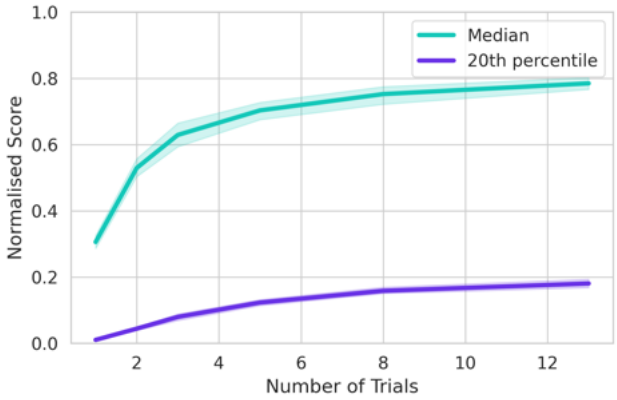


Figure I.3: 95% bootstrap confidence intervals around the median and 20th percentile over our test set of 5 agents trained with different initialisation seeds. The maximum observed standard deviation was 0.04 around the median and 0.02 around the 20th percentile.

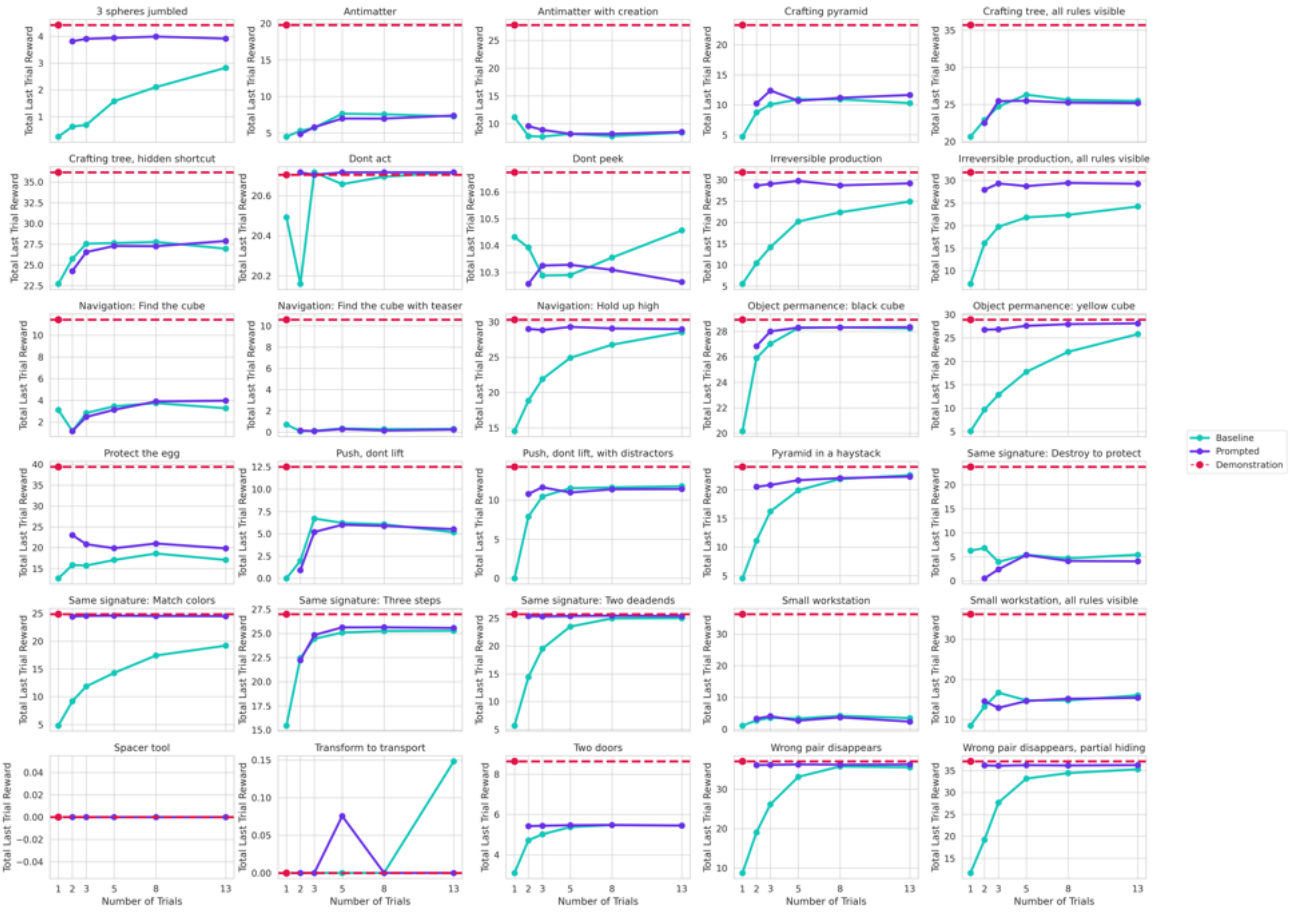


Figure I.4: A comparison of a prompted agent and unprompted baseline for our full set of 30 hand-authored single-agent tasks. The dashed red lines indicate the score obtained by the teacher providing the first-person demonstration to the prompted agent in the first trial.

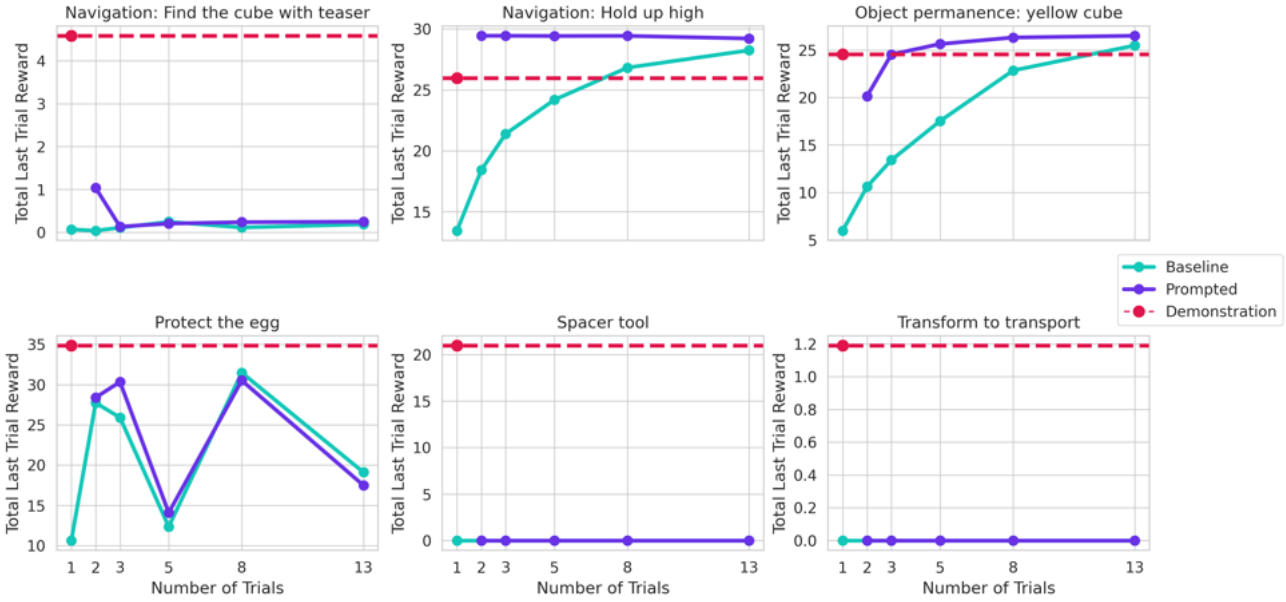


Figure I.5: Performance of AdA on 6 hand-authored tasks when prompted with an expert first-person human demonstration, compared with an unprompted baseline.

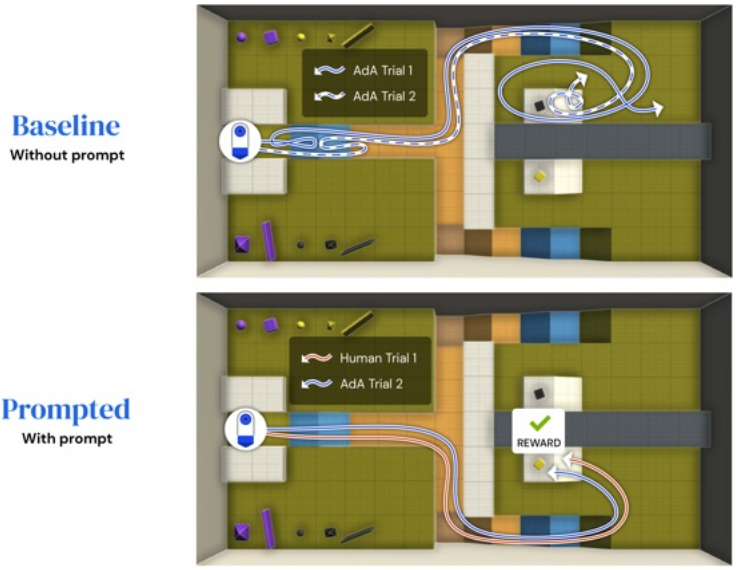


Figure I.6: Top-down views depicting the behaviour of AdA with and without a human expert prompt on the task `Object permanence: yellow cube`. On its own, AdA appears to have a bias of navigating to the black cube which is a dead end in this task. When prompted with a human (or fine-tuned) expert trajectory, AdA is able to overcome this bias and navigate to the yellow cube in the second trial.