PNAS RESEARCH ARTICLE | PSYCHOLOGICAL AND COGNITIVE SCIENCES
COMPUTER SCIENCES

Preemptive solving of future problems: Multitask Preplay in humans and machines

Wilka Carvalho[a,1], Sam Hall-McMaster[b], Honglak Lee[c,d], and Samuel J. Gershman[a,b]

Affiliations are included on p. 11.

Edited by Leslie Kaelbling, Massachusetts Institute of Technology, Cambridge, MA; received August 14, 2025; accepted August 13, 2026, by Editorial Board Member Terrence J. Sejnowski

Humans can pursue a near-infinite variety of tasks, but typically can only pursue a small number at the same time. We hypothesize that humans leverage experience on one task to preemptively learn solutions to other tasks that were accessible but not pursued. We formalize this idea as "Multitask Preplay", an algorithm that replays experience on one task as the starting point for "preplay"—counterfactual simulation of an accessible but unpursued task. Preplay is used to learn a predictive representation that can support fast, adaptive task performance later on. When implemented within deep reinforcement learning, it predicts human generalization to novel tasks more accurately than traditional planning and predictive representation methods across grid-worlds and a 2D Minecraft domain. Additionally, Multitask Preplay improves an artificial agent's ability to generalize to new environments sharing task co-occurrence structure. These findings demonstrate that Multitask Preplay is a scalable theory of human counterfactual learning and generalization; endowing artificial agents with this capacity can significantly improve their performance in challenging multitask environments.

cognitive science | counterfactual reasoning | deep reinforcement learning | transfer learning | naturalistic

Significance

How do humans leverage past experience to continually solve new tasks in a complex world? Inspired by hippocampal preplay where "place cells" activate before an animal pursues a location, we propose Multitask Preplay. Multitask Preplay preemptively simulates counterfactual pursuit of accessible but unpursued tasks and stores the solutions for fast, reactive behavior later without slow planning. We show this is a rare example of an algorithm that both captures human patterns of multitask problem solving in naturalistic tasks, and also improves AI performance in those tasks. These findings may suggest future computational accounts of hippocampal preplay, while also informing future algorithms for how agents can efficiently learn many interdependent tasks in settings where real-world experience is expensive (e.g., robotics).

While searching for coffee in a new neighborhood, you may come across gyms, grocery stores, and parks, enabling you to quickly find these locations later when you need them. One theory of how humans achieve this form of generalization posits that they learn an internal model of the environment, representing what is where (1). "Model-based" algorithms can use this model to chart a path from the current location to any goal. Model-based approaches provide maximum flexibility (generalization to arbitrary goals and starting points), but come at the cost of requiring a possibly prohibitive amount of computation at decision-time.

At the other extreme, "model-free" approaches directly learn predictions of future states or rewards based on interacting with the environment (2). For example, model-free deep reinforcement learning (RL) systems modify the weights of a neural network that outputs predictions of future reward based on sensory input (3). Predictions are effectively "cached" in the sense that they can be retrieved with relatively little additional computation. By choosing the actions that maximize future reward, a near-optimal policy can be cheaply computed. However, this comes at the cost of flexibility: a change in the task can require learning new weights from on-task experience.

In between these extremes are approaches that use offline computation ("background planning") to compile model-based knowledge into cached predictions (4, 5). This combines the flexibility of model-based approaches with the computational efficiency of model-free approaches. Classically, such approaches only cache predictions for a single task. This means that they are limited in their ability to flexibly generalize to new tasks without additional on-task experience, where each new task can require significant additional computation. Generalization is still bottlenecked by the need to rerun planning (either at decision-time or in the background) for each new task.

In fact, it is not necessary to run planning from scratch when different tasks share structure. For example, your search for coffee may lead you to a city center where many different shops are located; navigating to any of these potential goals will entail navigating to the city center and you can leverage this experience to learn about each of these goals. This shared structure can be leveraged by parameterizing a reward-predicting neural network to take the goal as input [what we refer to as "goal conditioning"; (6)]. However, learning to predict which actions will accomplish each goal still needs experience achieving that goal. Inspired by an old idea in RL that one can use experience with one goal to learn about many goals (7), we hypothesize that people preemptively gain this experience

Author contributions: W.C., S.H.-M., H.L., and S.J.G. designed research; W.C. performed research; W.C. contributed new reagents/analytic tools; W.C. analyzed data; and W.C., S.H.-M., H.L., and S.J.G. wrote the paper.

Competing interest statement: H.L. works at LG AI Research.

This article is a PNAS Direct Submission. L.K. is a guest editor invited by the Editorial Board.

Copyright © 2026 the Author(s). Published by PNAS. This article is distributed under Creative Commons Attribution-NonCommercial-NoDerivatives License 4.0 (CC BY-NC-ND).

[1]To whom correspondence may be addressed. Email: wilka@humanagencylab.io..

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2521870123/-/DCSupplemental.

Published November 19, 2026.

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

by counterfactually simulating pursuit of locally accessible goals and providing this simulated experience as input to a learning algorithm. Doing so then enables people to generalize to these goals later on with relatively little additional computation. We refer to this as Multitask Preplay.

In this paper, we describe Multitask Preplay formally and present a series of human behavioral experiments testing its distinctive predictions. In particular, we present evidence that humans counterfactually simulate pursuit of alternative goals, using these simulations to update cached predictions. We demonstrate the generalizability of Multitask Preplay by showing that our findings hold in both simple grid-worlds and in a complex, partially observable 2D Minecraft domain with numerous potential goals. We conclude with AI simulation results that show that Multitask Preplay improves an artificial agent's ability to generalize to 10,000 unique new environments with many hierarchical and interdependent tasks. Taken together, these results suggest that preemptive counterfactual learning is a powerful algorithmic concept for both natural and artificial intelligence.

Preemptively Solving Future Tasks With Multitask Preplay

We formalize Multitask Preplay within the framework of sequential decision problems. At any given time t, the environment is described by a state s_t. The agent selects an action a_t according to a probabilistic, state-dependent policy $\pi(a_t|s_t)$, and then receives reward r_{t+1}. The optimal policy π^* maximizes expected discounted future return, or *value*, defined as

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots | s_0 = s, a \sim \pi\right], \quad [1]$$

where $\gamma \in [0,1)$ is a discount factor determining the relative weighting of long-term vs. short-term rewards. The agent can potentially have multiple goals, but is constrained to pursue them sequentially. We use $R_g(s)$ to denote the goal-dependent reward function. Accordingly, we use $Q(s,a,g)$ to express the fact that different tasks induce different value functions, which can be incorporated into a single "universal" value function (6).

Value functions are interesting because they can be written recursively. This enables using a form of temporal difference (TD) learning known as Q-learning to estimate it (see *Materials and Methods* for more details). The TD loss function for a single transition is

$$L_{\mathtt{Q}\lambda} = \Big(\underbrace{r_t + \gamma \max_a \hat{Q}^{\pi}_{\theta'}(s_{t+1},a,g)}_{\text{target}} - \underbrace{\hat{Q}^{\pi}_{\theta}(s_t,a_t,g)}_{\text{prediction}}\Big)^2, \quad [2]$$

where $\hat{Q}^{\pi}_{\theta}(s_t,a_t,g)$ is an approximation of the value function parameterized by θ. We use deep neural networks to approximate the value function, updating the parameters by backpropagating the gradient of the TD loss through the network.

Parameterizing the neural network as a function of g allows its internal layers to learn shared structure across multiple tasks (6), and thereby generalize to a new task with goal $g_{\mathtt{new}}$. For example, it might be many steps from your house to the coffee shop downtown, but once downtown, it might only be a few steps off your trajectory to the grocery store; much of the value function is the same for both goals. While there is some evidence that humans use this strategy (8), nontrivial generalization in complex environments requires experience with millions of tasks (9). The bottleneck is direct experience: a neural network trained with TD learning can only generate good predictions in parts of the state space that are familiar to what the agent has explored before (Fig. 2*A*).

The same bottleneck applies to other methods. For example, the value function can be decomposed into a dot product between a reward vector $w_g(s)$ and "successor features" (SFs):

$$\psi^{\pi}(s,a) = \mathbb{E}\left[\phi(s_1) + \gamma\phi(s_2) + \gamma^2\phi(s_3) + \ldots | s_0 = s, a \sim \pi\right], \quad [3]$$

the expected future occurrence of features defined by $\phi(s)$ (10). The value function for a new goal $g_{\mathtt{new}}$ can then be approximated by taking the dot product between $\psi(s,a)$ and $w_{g_{\mathtt{new}}}(s)$ (11). As for the value function, SFs can be estimated with a function approximator $\hat{\psi}^{\pi}_{\theta}$ such as a neural network. SFs have been successful at explaining some aspects of human generalization (8, 12). However, it too can only generate predictions for familiar parts of the state space (Fig. 2*B*).

While agents might not have direct experience traversing all goal-relevant parts of the state space, they might be able to quickly synthesize a transition model. Think about what happens when you look at a map: even if you have not traversed the area covered by the map, you can imagine such traversals. This opens the door to forms of offline model-based computation for estimating values in the absence of direct experience. A classic version of this approach is the value iteration algorithm, which updates the values of all state-action pairs using dynamic programming. This is generally intractable for large state spaces, which is why most modern model-based algorithms rely on various forms of tree search, estimating values locally by simulating paths forward from the current state. Because these algorithms are applied at decision-time, they do not make use of offline computation. An alternative that mixes these ideas is Dyna (4), which uses TD learning to update value estimates locally around replayed past experiences, consistent with evidence from humans (5, 13). However, this approach is still limited by the experience bottleneck (Fig. 2*C*).

We can uncork the bottleneck by using a model to simulate counterfactual experience—in particular, the pursuit of accessible off-task goals. The intuition is that these are goals which are likely to be pursued in the future, and therefore the agent can benefit from preemptively "preplaying" pursuit of them, caching the results of this offline computation in the neural network parameters (Fig. 2*D*). This is the central idea underlying the Multitask Preplay algorithm (Fig. 1), described more formally in *Materials and Methods*. This algorithm combines and extends important aspects from the previously discussed approaches: it uses TD learning to efficiently train a deep neural network across accessible goals, with a goal-dependent parameterization to learn shared structure across goals, and uses offline simulation to estimate values for parts of the state space not directly experienced. While here we use Multitask Preplay to learn a value function, it could also be used to learn any predictive representation (e.g., successor features).

Evidence That Humans Use Counterfactual Simulation to Preemptively Solve New Tasks

Multitask Preplay posits that people 1) "preplay" sequences which accomplish counterfactual tasks accessible from their direct experience, and 2) cache the results of this computation into a predictive representation using RL.This model makes 4 predictions about how people will pursue novel goals that were

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

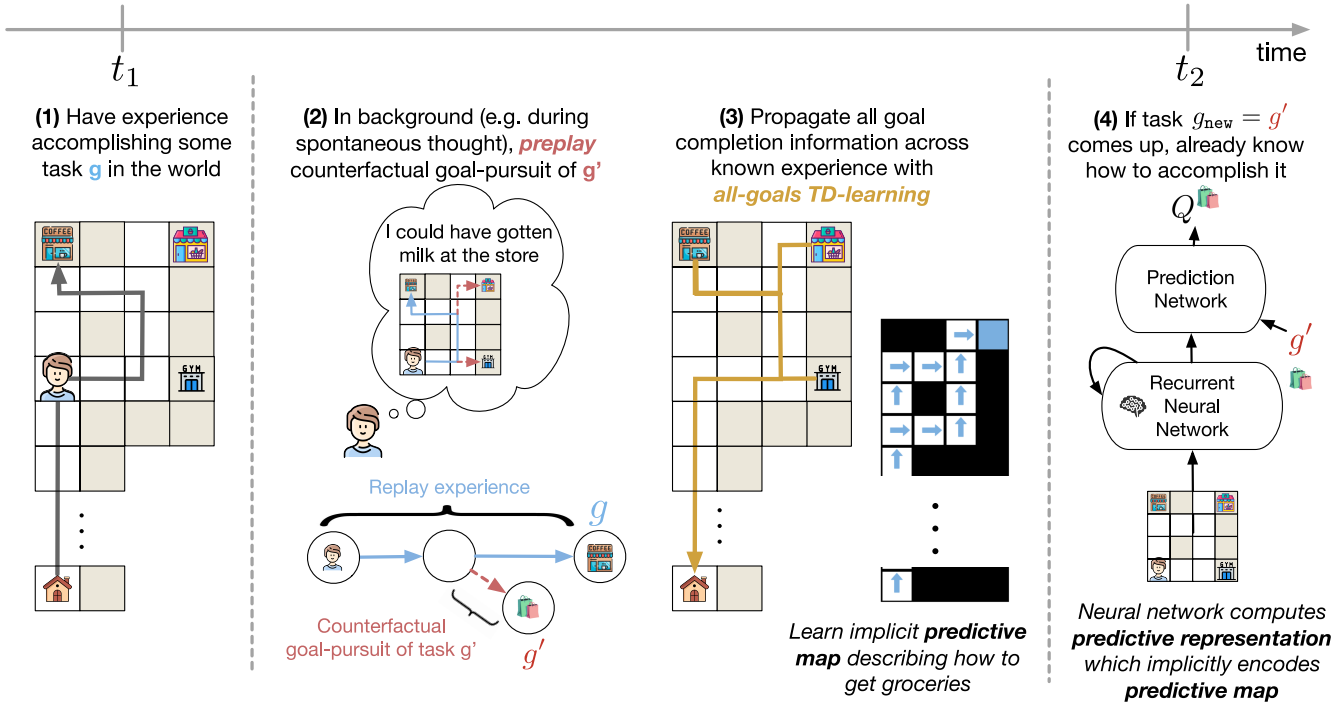

Fig. 1. Overview of Multitask Preplay. (1) At time t_1, an agent experiences a goal-directed task g (e.g., going to a coffee shop). (2) During offline replay, the agent counterfactually simulates alternative accessible tasks g' (e.g., going to a grocery store) based on the same experience trajectory. (3) Using temporal difference learning, the agent backs up information about both g and g' to learn a shared predictive representation. This enables it to form an implicit predictive map describing how to reach multiple goals. (4) At a future time t_2, if goal g' becomes the actual task, the agent can perform it reusing the previously cached predictive representation—without requiring new experience or planning. This architecture allows generalization to unseen but structurally accessible tasks through preemptive counterfactual simulation.

accessible from their prior experience. We tested each prediction in separate behavioral experiments using simple grid-worlds (see *Materials and Methods* for more details). Human subjects controlled a red triangle, which could move through the grid-world and consume food items. Subjects were first exposed to a set of training tasks, and then given one or more evaluation tasks designed to test their generalization. We collected $n = 100$ subjects for each experiment. Following past work, we use response times as a proxy for how much computation someone has performed (8, 14–16). All analyses were preregistered, except where noted. We note that the exact functions for computing path reuse were not preregistered, though the statistical tests we used were.

Prediction 1: Partial Reuse of Past Paths to Solve New Tasks. Past work has shown that people partially reuse past paths when solving new tasks in a small state space (17). We extend this work with experiment 1, by measuring reuse in a larger grid-world domain, exploiting response time data to test whether people are taking actions based on "preplayed" task-completion knowledge (Fig. 3*A*). If people are partially reusing paths, they should follow these paths for new tasks even when doing so is suboptimal (in terms of shortest path length). If they are relying on cached predictions rather than planning, they should be faster when reusing paths even though the path length is longer. Subjects were consistently successful at solving the evaluation task (Fig. 3*B*), a nontrivial achievement given that several sophisticated algorithms that learn generalizable predictive maps of the environment, including Universal Value Functions, Universal Successor Features, and HER (discussed further below), have generalization success rates near 0. Critically, subjects partially reused a training path a mean of 62.39% (95% CI: 53.99, 69.53) of the time, significantly greater than 50% ($P = 0.0049$, $Z = -0.15$, one-sided Wilcoxon signed-rank test; Fig. 3*B*; see *SI Appendix*, Fig. S1 for the path reuse distribution). Our preregistered analysis of the first response time (RT), which we expected to be the most diagnostic of people planning an entirely new path or reusing cached knowledge, showed that RTs were significantly lower on the first step of reused paths compared to new paths (linear mixed effects model: β=−0.21, 95% CI: −0.36, −0.06, t(133) = −2.81, P = 0.005; Fig. 3*C*). While most trials had a single RT significantly larger than all others when presumably most of the planning was happening (e.g., 1 s vs. 0.25 s), sometimes this occurred on later steps, somewhat idiosyncratically for different subjects (see *SI Appendix*, Fig. S2 for illustrative examples and *SI Appendix*, Fig. S3 for an analysis of RT magnitudes). We therefore conducted an exploratory follow-up analysis of the maximum RT on each path and identified a significant difference between partially reused and new paths (linear mixed effects model: $\beta = -0.32$, 95% CI: −0.49, −0.15, t(133) = −3.64, $P < 0.001$; Fig. 3*D*). Under the premise that longer RTs indicate longer planning operations, the two RT analyses provide converging evidence in support of the hypothesis that planning is faster with reuse, even when the paths are longer (see *SI Appendix*, Fig. S2 for some illustrative examples).

Most algorithms we tested (see *Materials and Methods* for details) either failed to match human success rates, or failed to match human path reuse rates (Fig. 3*B*; see *SI Appendix*, Fig. S4 for train performance). While Universal value functions can in principle generalize to new goals, they require experience in the relevant part of the state space (which our experiments precluded). Universal landmark successor features (SFs) are predictive representations that describe whether counterfactual goals are nearby or can be completed. They can be used to compute action values for new goals, but, unlike Multitask Preplay,

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

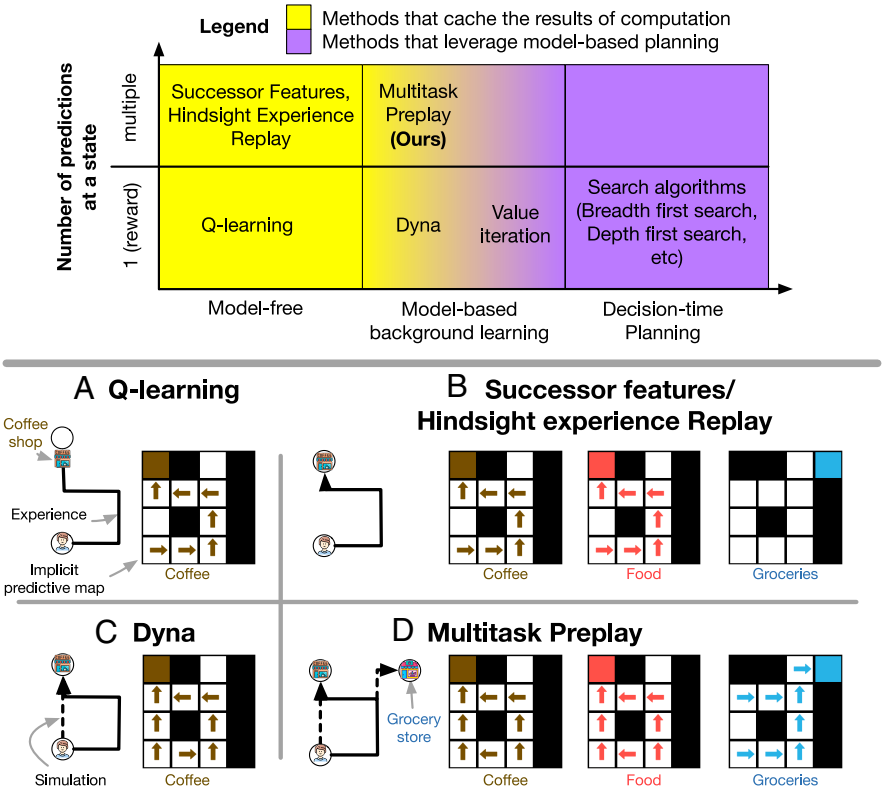

Fig. 2. Overview of algorithms. (*A*) Q-learning is a model-free algorithm that updates cached reward predictions based on interactions with the environment. (*B*) Successor features are predictive representations that generalize reward-predictive representations to arbitrary features. Like Hindsight Experience Replay (HER), they enable experience gathered while pursuing one goal to inform learning about other goals encountered along the way. Notably, if certain features (such as whether groceries are obtained) are not experienced, they are not encoded in the prediction. (*C*) Dyna uses model-based background learning to improve its cached reward predictions. (*D*) Multitask Preplay sits between Dyna and successor features. By leveraging counterfactual simulation of accessible tasks, it can learn to predict which actions lead not only to both the food and coffee, but also to the groceries.

value computation relies entirely on caching (no planning). This approach fails to generalize in our task because planning is required to chart action sequences off the training paths (see *SI Appendix*, Fig. S5 for analysis). HER similarly fails because its value estimates only cover goals along paths the agent has already traversed (see *SI Appendix*, Fig. S6 for analysis). Decision-time planning algorithms (breadth-first search and depth-first search) are able to chart such sequences, and hence achieve success rates comparable to humans, but they predict less reuse of the optimal training path compared to humans, and they do not predict shorter RTs for reused paths (planning time only depends on path length). However, we find that a Memory-based Euclidean Heuristic baseline (*Materials and Methods*) where a policy greedily shrinks Euclidean distance to the goal with a bias for reusing a remembered training path makes the same predictions as Multitask Preplay—i.e., a high success rate and path reuse while also predicting lower response times. We adjudicate between these two models in experiments 4 and 5.

Prediction 2: Partial Reuse of Past Paths to Solve New Tasks, Even When a Shortcut Opens Up. If people have preplayed a task that was previously accessible, they might forego planning when a shortcut opens up and partially reuse a prior path to the novel task (optimal length = 61, turns = 20). In contrast, if they have not, then they would plan at decision-time and more likely use the shortcut path that has an optimal length of 55 and 17 turns. We test for this in experiment 2 (Fig. 3*E*). Subjects partially reused a training path a mean of 62.80% (95% CI: 59.68, 73.91), significantly greater than 50% (one-sided Wilcoxon signed-rank test: $P < 0.001$, $Z = 1.08$; Fig. 3 *B* and *F*; see *SI Appendix*, Fig. S7 for the path reuse distribution). Like in our previous experiment, caching-based solutions fail to generalize: Universal Value Functions achieve 0% success, while mechanisms for transfer across goals fare only marginally better (Universal Landmark SFs 7.65%, HER 5.56%). Decision-time planning algorithms can form plans to accomplish these goals but predict significantly less path reuse than what people exhibit (39.00%, 95% CI: 31.50, 46.00 for depth-first search, 0% for breadth-first search). In contrast, Multitask Preplay, which predicts that people chart a path off their prior experience and cache the results, reproduces the human pattern of solving the task while preferentially reusing prior paths, with a mean success rate of 97.74% (95% CI: 96.71, 98.75) and path reuse rate of 99.50% (95% CI: 98.74, 100.00). Like in experiment 1, this is consistent with a Memory-based Euclidean Heuristic.

Prediction 3: Fast Response Times for New Tasks in Novel Starting Locations. In experiment 3, we look for evidence of task preplay at novel starting locations from a subject's prior experience. We use an evaluation protocol where a subject either begins at an intermediary position of their prior experience (condition 1) or in a novel part of the map (condition 2). Importantly, the optimal path in condition 2 (length = 44, turns = 14) is less than in condition 1 (length = 56, turns = 19; see Fig. 3*G*). This means that if they are planning in both conditions, they should be faster in condition 2. In contrast, if they have preplayed the novel task and cached the results already during training, they should be faster in condition 1, despite

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

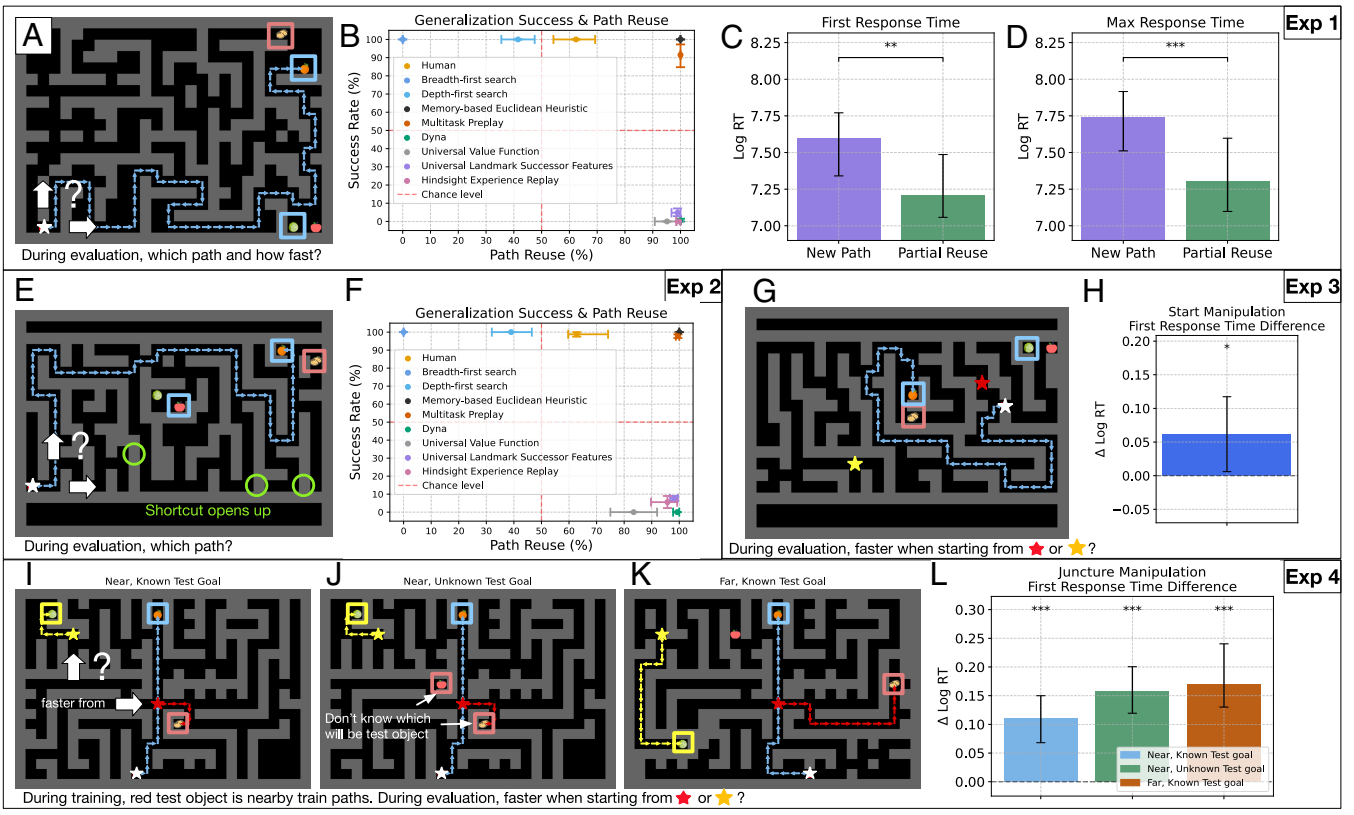

Fig. 3. Grid-world experiments and model comparisons. Across experiments, subjects learn to obtain training objects (blue boxes) before being evaluated on test objects (red/yellow boxes). White stars indicate spawning locations present in both training and evaluation; red/yellow stars show novel evaluation spawning locations. All figures display optimal paths from starting positions to target objects. Objects are randomized across subjects, who experience 4 trials per maze with different 90° rotations. We compare mean subject and model performance across these trials. All plots show the data mean and 95% CI. Experiment 1 (*A*–*D*) tests if people "preplay" the evaluation task as they complete the nearby training task (*A*; objects in top-right corner). (*B*) Model comparisons show only decision-time planning methods and Multitask Preplay match human success rates. Human path reuse rates fall between decision-time planning (Depth-first search) and Multitask Preplay. (*C* and *D*) Log first and maximum per-step response times during evaluation grouped by whether subjects took a new path or partially reused a training path. Experiment 2 (*E* and *F*) tests if people continue to be biased to partially reuse a past path even when a shortcut becomes available (*E*; note blue path crossing maze boundaries). (*F*) Model comparisons show people best match generalization success and path reuse rates of Multitask Preplay. Experiment 3 (*G* and *H*) tests if people are faster at evaluation task 1 (blue) vs. evaluation task 2 (yellow) even if it is longer. This would happen if the task was "preplayed" during training. (*H*) Log first response time difference between evaluation tasks 2 and 1. Experiment 4 (*I*–*L*) tests if people "preplay" evaluation task 1 (blue) at the closest location of a training path near the task object (blue star). In I and K, subjects know the blue evaluation goal during training. In J, it is unknown and one is randomly chosen from the two blue options. For all three, optimal paths during blue and yellow evaluation trials are equivalent but rotations of each other. (L) Log first response time difference between evaluation tasks 2 and 1.

it requiring a longer task solution. We measured $\Delta\text{Log RT} = \log\text{FirstRT(cond2)} - \log\text{FirstRT(cond1)}$. Despite condition 1 being 33% longer than condition 2, ΔLog RT was significantly greater than 0, i.e., people are significantly faster to respond at a further known location (one-sided Wilcoxon signed-rank test: median $\Delta\text{Log RT} = 0.06, (95\%\ \text{CI: } 0.01, 0.12), P = 0.015$; see Fig. 3*H*). This provides evidence that people propagate counterfactual task completion knowledge to novel starting points which were intermediary points of their previous experience.

Prediction 4: Fast Selection of Novel Behaviors that Solve New Tasks. The previous results suggested that people use cached knowledge to quickly pursue novel goals through familiar paths. While Universal Landmark SFs do not predict the previous behavioral results, one possibility is that people use a similar landmark-oriented algorithm to navigate toward a goal and then start planning when at a "juncture point" that departs from their previous experience. This contrasts with the prediction of Multitask Preplay that people have already "preplayed" actions that depart from the juncture point and require no more planning. To test for evidence of this preplay, we spawn people either at a juncture point (condition 1) or in a new part of the map where they will need to perform decision-time planning because their previous experience cannot be reused (condition 2). If they have preplayed the task, they should be faster in condition 1 than in condition 2. If they are performing decision-time planning once at the juncture point, then the response times should be approximately equivalent in the two conditions.

We ran 3 preregistered experiments. In the first (Fig. 3*I*), g' is nearby and known—similar to experiment 1. In the second (Fig. 3*J*), g' is nearby but unknown—this tests how automatic preplay for tasks may be as people do not know they will be evaluated on them later on. In the third (Fig. 3*K*), g' is far and known—this tests the bounds of preplay and shows whether it is robust to longer distances. We measured $\Delta\text{Log RT} = \log\text{FirstRT(cond2)} - \log\text{FirstRT(cond1)}$. ΔLog RT was significantly larger than 0, even though the distance to the goal was the same across conditions (one-sided Wilcoxon signed-rank test: median $\Delta\text{Log RT} = 0.11, Z = 4.56, (95\%\ \text{CI: } 0.07, 0.15), P < 0.001$ when near and known; median $\Delta\text{Log RT} = 0.17, Z = 6.43, (95\%\ \text{CI: } 0.12, 0.20), P < 0.001$ when near and unknown; and Paired t-test when far and known: median $\Delta\text{Log RT} = 0.172, t(99) = 5.85, (95\%\ \text{CI: } 0.131, 0.240), P < 0.001$;

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

Fig. 3*L*). We show examples in *SI Appendix*, Fig. S8 where we see that subjects can have a larger gap between the two conditions. We emphasize that generalization tasks had no overlap with training paths, so fast performance must come from caching. A natural caching-based solution is a Memory-based Euclidean Heuristic. However, its per-step computation is a fixed cost, so it predicts equal response times at the juncture point and the novel location; whereas humans reliably exhibit faster response times at familiar junctures. Thus, we hypothesize that people cache results of preplaying paths from juncture points—even when future goals are unknown.

Signatures of Multitask Preplay in a 2D Minecraft Domain

Here, we test Multitask Preplay's potential as a generalizable learning theory that can predict human behavior in increasingly ecologically valid conditions. We accomplish this by generalizing prediction 1 from the previous section to a more complex 2D Minecraft domain "Craftax" (18), where a subject needs to navigate to goal objects with only partial observation of a large world with many objects. We additionally extend prediction 1 to the setting where people are not told what object they will be evaluated on.

Prediction 5: Partial Reuse of Past Paths to Solve New Tasks in a Partially Observable, Large World With Many Objects. This preregistered experiment conceptually replicated experiment 1 in the Craftax domain. The key design feature of this experiment is the existence of multiple paths to get to a test object, and a suboptimal path overlapping with an optimal training path. Importantly, during training we provide subjects with the full map (Fig. 4*A*) and a partial view of the world (Fig. 4*B*) to mitigate the challenge of exploring this environment from only a partial view; however, during testing they only get the partial view. We construct maps such that test objects are visible when subjects get near a train object. We tell subjects that they are a crafter in a mining world with certain stones to mine. In condition 1, we tell subjects which will be their test stone goal, and in condition 2, replicating experiment 2, we do not tell them. This means that people are required to cache enough knowledge of this large map to support selecting actions to a new goal from a partial view of the world—like the real world.

In this more challenging domain, subjects consistently solve the evaluation task from only partial views of a large world. Both when subjects knew their future evaluation goal (condition 1) and when they did not (condition 2), they reused training paths to novel task objects a mean of 67.1% of the time in condition 1 (95% CI: [62.25, 72.00]) and 67.3% in condition 2 (95% CI: [62.33, 72.33]), significantly greater than 50% (one-sided Wilcoxon signed-rank test: $P < 0.001$ for both conditions; Fig. 4*C*; see *SI Appendix*, Fig. S15 for train performance; see *SI Appendix*, Fig. S16 for the path reuse distribution). In terms of decision-time planning approaches, no planning model we know of supports planning to novel goal objects from partial observations of a large world without assuming knowledge of the environment state space, so we do not compare to this model here. In contrast, approaches that rely entirely on caching fall short: Universal Value Functions and Universal Landmark SFs achieve 0% success, and HER reaches only 8.22%. The Memory-based Euclidean Heuristic, which reproduced human success and above-chance path reuse in experiments 1 and 2, requires exact knowledge of the goal's coordinates and of the remembered route—inputs that partial observability no longer supplies. When we model this with even mild uncertainty (goal location sampled within a radius of 2; remembered states dropped with probability 0.1), it maintains a high mean path reuse of 92.50% (95% CI: 88.33, 96.67) but its success tops out at 60.00% (95% CI: 51.67, 67.50; see *SI Appendix*, Fig. S32), well below human performance. In contrast, Multitask Preplay achieves a mean success rate of 97.5% (95% CI: 94.25, 99.50) and reuses prior paths 99.3% of the time (95% CI: 98.25, 100.00). Thus, Multitask Preplay provides a parsimonious account for what can support this kind of generalization in humans: learning a goal-oriented predictive map over many goals via counterfactual simulation.

Multitask Preplay Improves Generalization Performance of Artificial Agents in Novel Environments

So far, we have shown that when tasks co-occur, this can help with multitask generalization, as the solution for one task can be used to preemptively learn the solutions to other tasks. However, we have only considered generalization across tasks defined by acquiring a single object; whereas real-world tasks are often hierarchical with shared subtasks (e.g., you must get a knife before either slicing fruit or vegetables). Additionally, we often generalize known task behaviors to new environments (e.g., we can find and use knives when traveling to new homes). Here, we generalize our AI simulation results to Craftax (18). Craftax is a challenging long-horizon domain with a maximum score of 226 where 1) there is only a single hierarchical task composed of many interdependent subtasks, and 2) there are regularities in subtask co-occurrence. We argue that subtask co-occurrence can aid in multitask generalization across environments and study the extent to which Multitask Preplay can exploit this structure.

Prediction 6: Improved Generalization to Novel Environments that Share Subtask Cooccurrence Structure. We test this prediction in an augmented variant of the standard Craftax domain where we additionally expose agents to a function $\phi_{\texttt{avail}}(s)$ that describes what goals or subtasks are available in a state. Importantly, Craftax is a single-task domain where there is a single global reward function $R_g(s)$ that gives different priorities to different "subtasks" $\{g'\}$. By design, the reward function is simply additive: $R_g(s) = \sum_{g'} R_{g'}(s)$. We focus on how methods in Fig. 6 can improve generalization of an action-value function $\hat{Q}_\theta(s, a, g)$ and study how different methods can leverage $\phi_{\texttt{avail}}(s)$ to best enable generalization to 10,000 new unique environments after a variable number of training environments. We describe this protocol in more detail in *Materials and Methods*: Craftax Environment Transfer Details. Multitask Preplay leverages $\phi_{\texttt{avail}}(s)$ to sample $\{g'\}$ and then performs counterfactual simulation that optimizes for the corresponding $\{R_{g'}\}$. In contrast, Dyna simply uses $\phi_{\texttt{avail}}(s)$ as input for predicting $\hat{Q}_\theta(s, a, g)$ but performs counterfactual simulation that tries to optimize R_g for an environment. We have two model-free baselines `QL+1-step` and `QL+SF` which both use $\phi_{\texttt{avail}}(s)$ as either 1-step or n-step prediction targets to drive representation learning—a strategy which has been shown to improve generalization (19). Importantly, we keep the total number of training steps constant to 1 million steps for model-based background learning methods and 10 million steps for model-free methods.

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

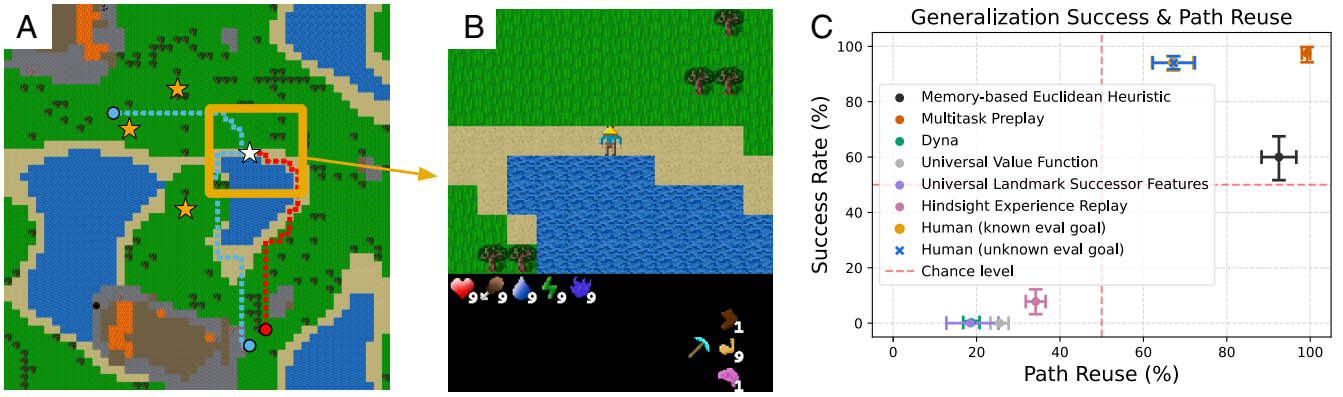

Fig. 4. 2D minecraft experiments and model comparisons. Subjects learn to obtain stones (a ruby, sapphire, or diamond) across 4 procedurally generated maps. Two are train stones, 1 is an evaluation stone. In one condition, they are told the evaluation stone; in another, they are not. (*A*) An example of the full map. Orange stars indicate spawning locations during training; the white star was a spawn location during both training and evaluation. We show the optimal path to train objects (blue) and evaluation objects (red). The evaluation object is always visibly nearby one of the training objects. (*B*) A zoomed-in view around the agent. During training, subjects get the full map and local view; during evaluation, they only get the local view. (*C*) Model comparisons show people best match generalization success and path reuse rates of Multitask Preplay. All plots show the data mean and 95% CI. Note that several points are overlapping and therefore not visible.

We find that as the number of training environments increases but the total steps to learn from stays fixed, Multitask Preplay best generalizes to 10,000 unique new environments, whereas Dyna and model-free methods asymptote in their generalization performance below this level (Fig. 5*A*). We suspect HER offers no advantage here because Craftax already rewards every reward-relevant goal a trajectory encounters, leaving nothing for relabeling to recover. Unsurprisingly, if we remove regularities in task object co-occurrence, we see a drop in performance (*SI Appendix*, Fig. S23). If we look at individual tasks, we see that Multitask Preplay can better perform tasks with more subtasks (Fig. 5*B*). That Multitask Preplay leads to improved training performance is unsurprising (*SI Appendix*, Fig. S24): since (20), it has been known that treating visible objects as goals can lead to faster learning on long-horizon tasks. However, what has not been shown before is that backing up subgoal g' knowledge to a global $\hat{Q}_\theta(s, a, g)$ over many environments that share subtask co-occurrence structure gives an effective "prior" Q-function with good generalization to new environments. We show detailed improvements across training settings in *SI Appendix*, Fig. S25 (for a comparison to state-of-the-art methods in the standard Craftax setting, see *SI Appendix*, Fig. S26). We emphasize that this is true even when keeping the number of training samples constant, leading to fewer training steps per environment (e.g., ≈2,000 steps/env when using 512 environments).

Discussion

Our experiments provide indirect evidence that people use counterfactual simulation to learn preemptively about off-task goals. People were faster at solving new tasks when they could partially reuse old solutions, even though those solutions were

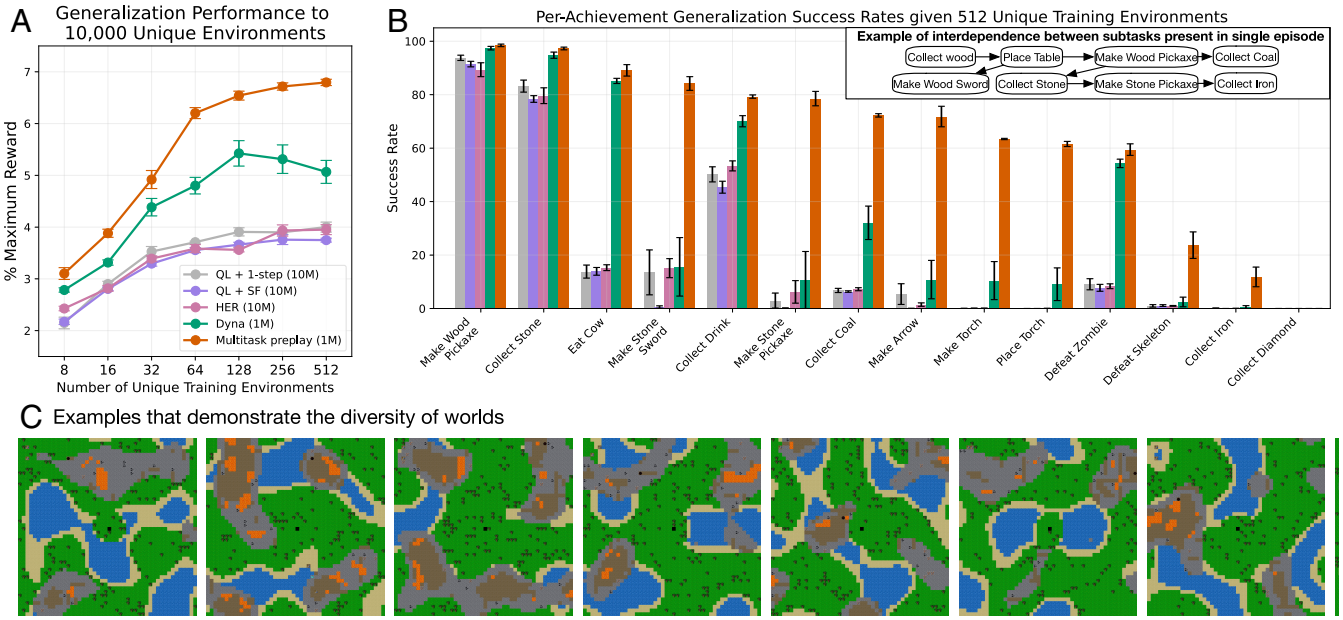

Fig. 5. AI simulation results studying generalization to 10,000 unique new testing environments. (*A*) Each point is the mean and SE across 5 model initializations (140 total individual training runs). Model-based methods are run for 1 million training steps and model-free methods are run for 10 million training steps. (*B*) Performance on individual subtasks during generalization when trained with 512 training environments. (*C*) Examples of procedurally generated maps in Craftax domain. We hold-out 10,000 unique maps for evaluation.

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

longer than the optimal path. This form of reuse arises specifically when people can generate counterfactual plans to off-task goals from their current position. People exhibited a tendency to reuse past solutions even when better paths became available. Similar patterns were observed in a more complex, partially observable domain (a 2D version of Minecraft).

We argue that this reuse behavior is adaptive, despite introducing systematic biases, because it enables people to achieve flexibility to solve many different tasks without significant decision-time planning—a fundamental bottleneck for model-based algorithms. Across machine learning, cognitive science, and neuroscience, Universal Value Functions and Successor Features have been proposed as alternative approaches that enable cheap computation of this kind of flexible behavior (6, 8, 12, 21). However, our simulation results show that this is not the case for the domains we study. In contrast, we show that a new algorithm, Multitask Preplay, succeeds where humans succeed—and accomplishes these successes with human-like biases for reusing previous solutions. The algorithm incorporates preemptive counterfactual simulation into a general and powerful RL architecture that can significantly improve generalization to new tasks compared to modern baselines. While each individual experiment may admit an idiosyncratic alternative theory—for example, people may avoid unknown paths as "risky" for experiment 1—Multitask Preplay provides a simple and parsimonious answer across the 5 experiments we ran. Another limitation of our work is that we currently provide only indirect evidence via response time correlations and somewhat coarse behavioral predictions. Future work can pursue complementary measurement tools such as eye tracking (as in ref. 22) or neuroimaging (as in ref. 13).

In addition to our cognitive science results, our AI simulations show that learning from both real and counterfactual tasks leads to more robust reward predictions, and improves generalization to 10,000 unique new environments that share task co-occurrence structure with the training environments. Unfortunately, this experiment would be hard to run with humans, as they would likely require prohibitively long sessions. We use the AI results to argue that this property common to real-world environments is something that can be effectively exploited by Multitask Preplay.

While past human behavioral studies have provided evidence for policy reuse (17) and offline simulation (5, 13), these experiments used environments with small state spaces and lacked generalization tests to novel goals. A number of studies have examined feature-based generalization (23–25), but only in bandit tasks that lack sequential structure. Other studies have examined generalization to novel goals (8, 26), but were not designed to test for counterfactual simulation. Ours is a series of experiments to comprehensively test the hypothesis that people engage in preemptive counterfactual simulation of off-task goals.

Within machine learning, a common strategy for learning about alternative goals is to leverage HER (27), which relabels failed experiences by setting reached states as counterfactual goals. This can accelerate main-task learning (28) and improve exploration (29, 30). While some work has used states from erroneous planning for relabeling (31), none intentionally selects available but unpursued goals for simulation. Multitask Preplay instead chooses unpursued goals and augments experience with counterfactual simulation toward these goals. Our approach falls within the "many-goals RL" framework that uses each experience to learn about multiple goals (7, 32). While prior work used model-based simulation for many goals in robotics (33), it focused on unsupervised pretraining rather than the orthogonal issue of integrating replay and counterfactual simulation to learn about other accessible but unpursued tasks. Kaelbling et al. (7) first described combining Dyna with counterfactual simulation for many goals, but neither they nor subsequent work demonstrated simulation results or explored transfer benefits. We show that this enables learning of goal-oriented predictive representations that transfer to new tasks and new environments where tasks share co-occurrence structure, and detail the algorithmic components needed for successful model-based all-goals learning in large worlds.

Our work raises a number of questions. First, how do people select off-task goals? We studied only the simplest possible selection algorithm, but one could sample goals based on criteria such as uncertainty, future relevance, or learning progress. This may allow agents to better allocate computational resources as the number of possible goals increases in naturalistic environments. Second, our simulations assumed access to an oracle world model, but in general this needs to be learned from experience. Future work could either leverage a Vision-language model such as RT-2 (34), or learn the model with a modern algorithm such as MuZero (35). Finally, we have only studied video game applications in this paper, but Multitask Preplay could be applied to a wider range of challenging problems. For example, the Habitat AI domain (36, 37) contains virtual scans of thousands of real homes which possess the task co-occurrence property we argue Multitask Preplay exploits. This domain could provide an opportunity to compare human and AI performance on naturalistic object-oriented tasks, such as finding mugs or preparing food (38).

Our work additionally opens up interesting questions for neuroscience. Multitask Preplay is inspired by preplay activity in the hippocampus, where place cells (neurons selective for particular locations in space) are sequentially activated during sleep or quiescence. Preplay sequences are distinct from "replay" sequences in that they do not rigidly reflect an animal's past experience, but rather experience it has later once awake (39–41). The computational function of neural preplay is currently unclear, but it may play a role enabling fast, adaptive decision-making. Multitask Preplay may offer a new theoretical explanation, and future neurophysiology studies could use versions of our experimental protocols to more specifically test this hypothesis.

Materials and Methods

We formulate domains as Partially Observable Controlled Markov Processes $\mathcal{C} = \langle \mathcal{S}, \mathcal{A}, \mathcal{X}, P, O \rangle$, where $\mathcal{S}$ denotes the environment state space, $\mathcal{A}$ denotes its action space, and $\mathcal{X}$ denotes (potentially partial) observations of the environment. When an agent takes an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, the next state s' is sampled according to a next state distribution $s' \sim P(\cdot|s, a)$, and an observation x' is generated deterministically via $O(s')$. We formulate tasks as Partially Observable Markov Decision Processes $\mathcal{M} = \langle \mathcal{C}, R_g \rangle$, with goal-parameterized reward functions: $r_t = R_g(s_t)$. Henceforth, we will use "goal" and "task" interchangeably since a goal specifies a task in this setup. We train both humans and models using the same set of training tasks $\mathbb{M}_{\text{train}} = \{\mathcal{M}\}_{i=1}^{n_{\text{train}}}$ and then evaluate them on the same set of evaluation tasks $\mathbb{M}_{\text{test}} = \{\mathcal{M}\}_{i=1}^{n_{\text{test}}}$.

For all deep RL models, we assume access to a learned "agent state" representation which aggregates all information observed so far within an episode via a recurrent function of history (e.g., a recurrent neural network); to avoid proliferation of notation, we will assume here that the agent and environment states are identical, but in general they need not be. Thus, we write the recurrent state function as $s_t = f_\theta(x_t, s_{t-1}, a_{t-1})$, where θ are the model parameters. We optimize models so that they learn a behavior policy $\pi_\theta(a|s, g)$ that maximizes the future reward it can expect from taking an action a in state

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

s, i.e., to maximize the value function (Eq. **1**). All models use a target network (3) to stabilize learning. A separate copy of the parameters θ' is held fixed while computing bootstrap targets and periodically synchronized with θ every 1,000 environment steps, preventing the optimization targets from shifting with each gradient update.

All deep RL algorithms we consider have a "replay" component to training: after an agent experiences a trajectory $\tau = (x_1, a_1, \ldots, a_{T_{\text{env}}-1}, x_{T_{\text{env}}})$ in the environment accomplishing some task g, both are added to a memory buffer $\mathcal{B}$. For all learning algorithms, subsequences are then sampled and "replayed", i.e., $\tau \sim \mathcal{B}$, and processed by a history function to obtain agent state representations for each time-point $s_t = f_\theta(x_t, s_{t-1}, a_{t-1})$ leading to $\tau = (s_1, \ldots, s_T)$, where $T << T_{\text{env}}$. Algorithms then differ in how they construct a learning objective with these data.

Q-Learning. The simplest form of model-free RL is to directly approximate the value function (Eq. **1**) using a parameterized function approximator $\hat{Q}^\pi_\theta(s_t, a_t, g)$, here a deep neural network, trained to optimize a loss function comparing value estimates to a target function (Eq. **2**). The target function is a "bootstrapped" estimate of the true value function. The target in Eq. **2** is called a 1-step return because it combines the observed reward at one timestep, r_t, with the estimated return at subsequent timesteps, $\max_a \hat{Q}^\pi_{\theta'}(s_{t+1}, a, g)$, discounted by γ. Although the target relies on the value function approximator itself (this is the sense in which it is bootstrapped), it can nonetheless be used to incrementally improve the approximator.

In practice, it is advantageous to optimize a generalized target, the λ-return (42):

$$G^\lambda_t = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\left[r_t + \gamma r_{t+1} + \cdots \gamma^n \max_a \hat{Q}^\pi_{\theta'}(s_{t+n}, a, g)\right], \quad [4]$$

where $\lambda \in [0, 1]$ is a parameter that controls the exponential weighting of different n-step returns (the term in brackets). Setting $\lambda = 0$ recovers the original 1-step return. All settings of λ are valid targets for estimating the value function, but they realize different bias–variance trade-offs. In large state spaces where goals may be many steps away, high λ is critical: the 1-step return propagates reward information only one step per update, requiring many passes before signal reaches early states, whereas multistep returns propagate information across the full trajectory in a single update. We use $\lambda = 0.9$, which achieves good results for our applications. Using the λ-return as a target, the complete loss function becomes

$$L_{Q\lambda}(\tau, g) = \sum_{t=1}^{T-1}\left(G^\lambda_t - \hat{Q}^\pi_\theta(s_t, a_t, g)\right)^2, \quad [5]$$

where the single-transition loss is now averaged over all transitions in the replay buffer.

To map the value function to actions, we use a simple ϵ-greedy policy, which chooses $a_t = \text{argmax}_a \hat{Q}^\pi_\theta(s_t, a, g)$ with probability $1-\epsilon$ or a random action with probability ϵ. This ensures that the agent explores actions which are currently low-value but might turn out to be high-value with more experience. Much more sophisticated exploration policies have been developed, but our focus here is on learning algorithms, which is why we deliberately adopted the simplest possible policy.

Dyna. Q-learning relies purely on experienced trajectories for value estimation. If the agent additionally has access to a model of the environment, $\hat{P}$, it can synthesize ("preplay") data for improving its estimates (provided the model is reasonably accurate). This is the essential idea underlying the Dyna architecture (4), summarized in Fig. 6.

In our implementation of Dyna, synthetic data τ_{sim} is sampled from the environment model by running the model forward from observations in the replay buffer. The action selection policy is the same as the one used to generate real actions. This is repeated to produce n_{sim} trajectories stored in a preplay buffer. The synthetic data are treated in the same way as real data, entering into the TD loss (Eq. **5**).

Multitask Preplay. Multitask Preplay (summarized in Fig. 6) is a multitask variant of Dyna designed to leverage each collected trajectory for learning across the full range of an agent's goals, not just the one being pursued. The central idea builds on the distinction between "on-task and off-task" learning (21), which parallels the classical RL distinction between on-policy and off-policy learning. Off-policy learning uses data from one policy to improve another, requiring corrections because the behavior policy differs from the target. Off-task learning uses data collected for one goal to learn about a different goal, requiring corrections because trajectories for a data-generating goal often diverge from optimal trajectories for other goals.

Multitask Preplay exploits off-task learning through two mechanisms, "preplay" and "conservative all-goals learning". Preplay augments trajectories with counterfactual simulations of other goals, expanding the set of goals each trajectory can inform. Conservative all-goals learning reuses collected trajectories directly to update value estimates for goals beyond the one being pursued. Together, the two mechanisms allow every trajectory to contribute learning signal for all known goals. Both require off-task corrections, and our key insight is that existing off-policy machinery–specifically Peng's $Q(\lambda)$ (43)–can be repurposed for this purpose.

The challenge is that multistep returns yoke the trace to the actions actually taken. High λ is critical for propagating reward across long trajectories, but when those actions were chosen to optimize g, the λ-weighted return accumulates divergence from what is optimal for g' step by step. The λ-return (Eq. **4**) makes this concrete once rewritten recursively:

$$G^\lambda_t = r_{t+1} + \gamma\left[\lambda\, G^\lambda_{t+1} + (1-\lambda)\max_{a'}\hat{Q}^\pi_{\theta'}(s_{t+1}, a', g)\right], \quad [6]$$

with $G^\lambda_T = \max_{a'}\hat{Q}^\pi_{\theta'}(s_T, a', g)$ at the terminal step. Here λ acts as a per-step trace gate, which suggests a natural correction: modulate λ_t according to whether the collected action agrees with what the off-task goal would prescribe. We call this "off-task $Q(\lambda)$", repurposing the trace-cutting mechanism of Peng's $Q(\lambda)$ (43) from the policy axis to the goal axis

$$\lambda_t = \begin{cases} \lambda & \text{if } g = g' \\ \lambda \cdot \mathbf{1}\left[\text{argmax}_{a'}\hat{Q}^\pi_\theta(s_t, a', g') = a_t\right] & \text{if } g \neq g' \end{cases}. \quad [7]$$

When goals agree, the full multistep trace flows through and the correction reduces to standard $Q(\lambda)$. When they disagree, the trace is cut and the return falls back to a bootstrapped estimate. The off-task target G^{off}_t is obtained by substituting λ_t for the fixed λ in Eq. **6**, yielding:

$$L_{\text{OQ}\lambda}(\tau, g') = \sum_{t=1}^{T-1}\left(G^{\text{off}}_t - \hat{Q}^\pi_\theta(s_t, a_t, g')\right)^2. \quad [8]$$

With this correction in hand, preplay proceeds as follows. At each experienced state s_t, the agent samples n_{goals} counterfactual goals $G = \{g_i \sim p_{\text{sim}}(\cdot|s_t)\}_{i=1}^{n_{\text{goals}}}$ uniformly from accessible goals, then generates simulated trajectories as in Dyna–but under a mixed policy that interpolates between the current and counterfactual goals:

$$a_t = \text{argmax}_a\, \alpha_g \hat{Q}^\pi_\theta(s_t, a, g) + \alpha'_g \hat{Q}^\pi_\theta(s_t, a, g'), \quad [9]$$

where α_g and α'_g are the weights for the two goal-conditioned action-value functions. The interpolation biases simulation toward the counterfactual goal while allowing the trajectory to revert once that goal is completed. Each rollout yields learning signal for both g' and the original goal g via off-task $Q(\lambda)$.

Conservative all-goals learning addresses a complementary problem. Rather than simulating new trajectories, it reuses each collected trajectory τ directly, updating Q-values for goals g' sampled from a distribution $p_{\text{all}}(\cdot|s_t)$ over all known goals. Off-task $Q(\lambda)$ handles the trace correction, but a subtler issue remains: the greedy action for g' may be suboptimal for g, so the agent selects it less and less during on-task collection. Any overestimation of that action's Q-value becomes self-reinforcing–never tested, never corrected. We prevent this

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

Dyna	Multitask Preplay
Sample trajectory τ and goal g via replay	Sample trajectory τ and goal g via replay
// On-task simulation	*// Preplay*
For $t \in (1, \ldots, T)$	For $t \in (1, \ldots, T)$
For $i \in (1, \ldots, n_{\texttt{sim}})$	For $i \in (1, \ldots, n_{\texttt{preplay}})$
	For $j \in (1, \ldots, n_{\texttt{goals}})$
	Sample counterfactual goal: $g' \sim p_{\texttt{sim}}(\cdot \mid s_t)$
Sample trajectory $\tau_{\texttt{sim}}$ for g starting at s_t	Sample trajectory $\tau_{\texttt{sim}}$ for g' starting at s_t
Update loss: $L \leftarrow L + L_{\texttt{Q}\lambda}(\tau_{\texttt{sim}}, g)$	Update loss: $L \leftarrow L + L_{\texttt{OQ}\lambda}(\tau_{\texttt{sim}}, g)$
	Update loss : $L \leftarrow L + L_{\texttt{OQ}\lambda}(\tau_{\texttt{sim}}, g')$
// Single-goal learning	*// Conservative all-goals learning*
Evaluate loss: $L \leftarrow L_{\texttt{Q}\lambda}(\tau, g)$	For $j \in (1, \ldots, n_{\texttt{goals}})$
	Sample goal: $g' \sim p_{\texttt{all}}(\cdot \mid s_t)$
	Update loss: $L \leftarrow L + L_{\texttt{COQ}\lambda}(\tau, g')$
Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta L$	Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta L$

Fig. 6. Comparison between Dyna and Multitask Preplay. In our experiments, $n_{\text{sim}} = n_{\text{goals}} \cdot n_{\text{preplay}}$, so that Dyna and Multitask Preplay have the exact same simulation budget. The difference is in how simulations are chosen and how they are used for learning. Note that the goal distribution $p_{\text{sim}}(\cdot \mid s_t)$ includes the on-task goal g, so Multitask Preplay subsumes standard Dyna-style on-task simulation as a special case.

with a conservative regularizer that penalizes Q-values for actions absent from the collected data (44):

$$L_{\text{CL}}(\tau, g') = \sum_{t=1}^{T-1} \left[\log \sum_a \exp \hat{Q}^{\pi}_{\theta}(s_t, a, g') - \hat{Q}^{\pi}_{\theta}(s_t, a_t, g') \right]. \quad [10]$$

The complete loss, "Conservative Off-task $Q(\lambda)$", combines off-task $Q(\lambda)$ with the regularizer*:

$$L_{\text{COQ}\lambda}(\tau, g') = L_{\text{OQ}\lambda}(\tau, g') + \eta_C \cdot \mathbf{1}[g' \neq g] \cdot L_{\text{CL}}(\tau, g'). \quad [11]$$

Fig. 7 illustrates how $L_{\text{Q}\lambda}$, $L_{\text{OQ}\lambda}$, and $L_{\text{COQ}\lambda}$ differ in their value-backup behavior when learning off-task Q-values and *SI Appendix*, Fig. S27 ablates Eqs. **7**, **10**, and **11**.

Universal Successor Features. Successor features (SFs) are a form of predictive representation that generalize the standard reward-predictive representation (Eq. **1**) used in RL to arbitrary feature-predictive representations (2). The key assumption of SFs is that reward for goal g can be expressed as a dot-product between state-features $\phi(s)$ and a "preference vector" w_g for those features $R_g(s) = \phi(s)^\top w_g$, where $\phi(s), w_g \in \mathbb{R}^d$. SFs are defined as the predictions of the expected discounted accumulated features ϕ under policy π:

$$\psi^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_{t+1}) \mid s_0 = s, a \sim \pi \right]. \quad [12]$$

Action-values can be expressed as a dot-product between SFs and a corresponding preference vector: $Q^{\pi}(s, a, g) = \psi^{\pi}(s, a)^\top w_g$. To facilitate generalization across tasks and policies, we will consider Universal Successor Feature Approximators (21), which are the analogue of goal-conditioned value functions described above where the function approximator takes some reference to the policy for which it is estimating SFs, such as a task parameter g, as input $\hat{\psi}^{\pi}_{\theta}(s, a, g)$.

We assume that the features $\phi(s)$ are additionally defined by encounters with landmark objects. In fully observable domains such as JaxMaze (Fig. 3), $\phi_k(s) = 1$ every time landmark k is within some radius of the agent (0 otherwise). In partially observable domains such as Craftax (Fig. 4), landmark features activate when the object is visible to the agent.

The structure of the learning algorithm is essentially the same as for Q-learning, with the value function replaced by SFs. An analogous λ-return can be defined for SFs (again replacing rewards with features). Parameters of the function approximator are likewise optimized to the squared difference between the empirical and predicted λ-returns.

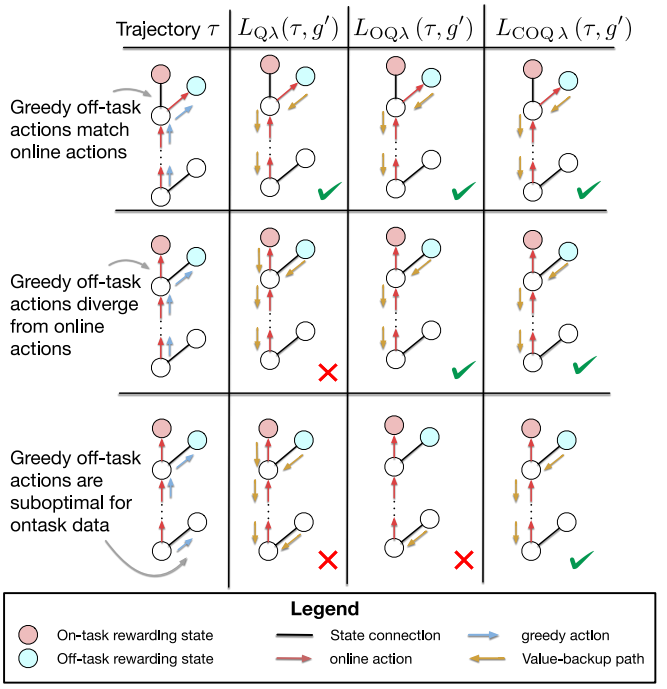

Fig. 7. Comparison of value-backup strategies for off-task learning. Each row depicts a trajectory (*Left*) and the value-backup paths under $Q(\lambda)$ ($L_{\text{Q}\lambda}$), off-task $Q(\lambda)$ ($L_{\text{OQ}\lambda}$), and conservative off-task $Q(\lambda)$ ($L_{\text{COQ}\lambda}$) (*Right* three columns). *Top*: When the greedy actions for the off-task goal match the collected actions, all three losses correctly propagate value from the rewarding state. *Middle*: When those greedy actions diverge from the collected actions, $Q(\lambda)$ erroneously backs up information from the on-task reward, which may be zero; off-task $Q(\lambda)$ and its conservative variant cut the trace and propagate only from the off-task rewarding state. *Bottom*: When the Q-value of an unseen action is overestimated, $Q(\lambda)$ backs up value from both the true reward and that inflated Q-value, while off-task $Q(\lambda)$ backs up from the inflated Q-value alone—compounding the error. Only the conservative variant, which penalizes Q-values for actions absent from the data, suppresses the inflated estimate while still propagating from the true reward.

*A similar loss was concurrently developed outside of the all-goals setting by Kim and Oh (45) for conservative value estimation during offline RL.

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

As with Multitask Preplay, we assume that the algorithm knows about all accessible goals within a domain $G = \{g_i\}_{i=1}^{n_{\text{train}}+n_{\text{test}}}$, along with their associated preference vectors. Thus, a trajectory τ collected for goal g could be used to learn from all goals by randomly sampling goals and optimizing the aggregated loss. While in principle this endows the algorithm with the same generalization capabilities as Multitask Preplay, it critically depends on the ability to directly experience many different tasks, so that all the heavy lifting can be done by a purely model-free learning algorithm. The premise of this paper is that agents in the real world rarely have access to such abundance of direct experience.

SFs also enable another form of generalization through the reuse of previous policies for a new task. The key algorithmic tool is generalized policy iteration (GPI) (11), which selects actions for a new task g_{new} according to:

$$a(s, g_{\text{new}}) = \underset{a\in\mathcal{A}}{\text{argmax}} \max_{w\in\mathcal{C}} \{\hat{\psi}_{\theta}^{\pi}(s, a, w)^{\top} w_{g_{\text{new}}}\}, \quad [13]$$

If $\mathcal{C} = \mathcal{W}_{\text{train}}$, then this corresponds to seeing if any training policy leads to features specified by $w_{g_{\text{new}}}$. However, if we learned $\hat{\psi}_{\theta}(s, a, w_{g_{\text{new}}})$ with "counterfactual task learning" (i.e., collected data with $\mathcal{W}_{\text{train}}$ but computed learning updates for $w_{g_{\text{new}}}$), we can see if $\hat{\psi}_{\theta}(s, a, w_{g_{\text{new}}})$ does not already contain landmark and task-completion knowledge that optimizes $w_{g_{\text{new}}}$. Two options for using this knowledge are to set $\mathcal{C} = \mathcal{W}_{\text{train}} \cup \{w_{g_{\text{new}}}\}$ or $\mathcal{C} = \{w_{g_{\text{new}}}\}$.

HER. HER (27) enables off-task learning without an environment model by relabeling collected trajectories with goals achieved along the way. When an agent collects a trajectory τ while pursuing goal g, HER identifies hindsight goals from states achieved in the trajectory and replays subtrajectories with these substituted goals, optimizing the standard $Q(\lambda)$ loss (Eq. **5**). This relabeling transforms failures into training signal: a trajectory that failed to reach g may have achieved other goals along the way. HER is primarily an exploration algorithm–by building Q-values for incidentally achieved goals, it constructs a richer value landscape that facilitates reaching the goal of interest. However, HER can only learn about goals g' that lie along the path the agent has taken. In large state spaces, where trajectories cover a diminishing fraction of the goal space, this limits HER's ability to propagate value information beyond directly visited goals–the same limitation faced by USFA.

HER With Conservative All-Goals Learning. TD bootstrapping can propagate value to goals absent from the current trajectory, suggesting a direct extension: augmenting HER with the conservative all-goals loss $L_{\text{COQ}\lambda}$. For each trajectory τ, goals g' are sampled from $p_{\text{all}}(\cdot|s_t)$, and the off-task $Q(\lambda)$ correction handles the goal mismatch while the conservative regularizer prevents overestimation. This extension enables HER to scale to large environments where standard HER plateaus, provided the evaluation goals lie along paths the agent has previously traversed (*SI Appendix*, Fig. S6). When goals are not along any experienced path, HER with all-goals learning faces the same limitation as USFA: without direct experience of relevant states, purely model-free methods cannot construct informative value estimates.

Euclidean-Distance Baselines. A final family of baselines chooses actions by how much they shrink the Euclidean distance to the goal. The simplest is a memory-less greedy Euclidean policy, which at each step takes the action that brings it closest to the goal,

$$a_t = \underset{a}{\text{argmax}} \left(d(s_t, g) - d(\hat{T}(s_t, a), g) \right), \quad [14]$$

where d is Euclidean distance and $\hat{T}(s_t, a)$ is the agent's internal world model, which predicts the next state $\hat{s}_{t+1}$ that follows from taking action a. Like with Dyna and Multitask Preplay, our experiments assume a perfect world model $\hat{T}(s_t, a)$. This policy descends the distance gradient to the goal, but it keeps no record of where it has previously traveled.

We can supply such a record by biasing actions toward previously taken paths. Let $M(g)$ be a memory of the states the agent occupied while previously pursuing goal g. The memory-based variant, which we call the "Memory-based Euclidean Heuristic", adds a bonus for reentering these states,

$$a_t = \underset{a}{\text{argmax}} \left(d(s_t, g) - d(\hat{T}(s_t, a), g) + \alpha \mathbf{1}\left[\hat{T}(s_t, a) \in M(g)\right] \right), \quad [15]$$

where α controls how strongly the agent prefers to follow a prior path. We use this heuristic to test how far distance minimization plus a path-reuse bonus can recapitulate our behavioral results (*SI Appendix*, Fig. S31). Main-text figures report the $\alpha = 10$ policy.

Both policies critically assume access to $d(s_t, g)$. In partially observable domains the agent is instead uncertain about the goal location and about which states belong to $M(g)$. We model the former by sampling the goal within a radius r of g, and the latter by dropping each remembered state from $M(g)$ with probability P (*SI Appendix*, Fig. S32). Main-text Craftax figures use the mild-uncertainty setting ($r = 2$, $P = 0.1$).

Ethics Statement. All experiments were approved by the Committee on the Use of Human Subjects at Harvard University. All subjects were recruited with CloudResearch and provided informed consent.

Data, Materials, and Software Availability. Data and code data have been deposited in 1) Open Science Foundation (https://doi.org/10.17605/OSF.IO/M53QH (46) and https://doi.org/10.17605/OSF.IO/B2EVM (47)) and 2) GitHub (https://github.com/wcarvalho/multitask_preplay) (48). All other data are included in the manuscript and/or *SI Appendix*.

ACKNOWLEDGMENTS. We thank Andrew Lampinen, Angelos Filos, Shuze Liu, Anne Collins, Tomer Ullman, and the Computational Cognitive Neuroscience Lab at Harvard generally for helpful discussions on the topic. This work was supported by the Department of Defense Multidisciplinary University Research Initiative program under Army Research Office Grant W911NF-23-1-0277, a Polymath Award from Schmidt Sciences, a Philip Wrightson Award from the New Zealand Neurological Foundation, and by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence.

Author affiliations: [a]Kempner Institute for the Study of Natural and Artificial Intelligence, Harvard University, Allston, MA 02134; [b]Department of Psychology and Center for Brain Science, Harvard University, Cambridge, MA 02138; [c]Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109; and [d]LG AI Research, Ann Arbor, MI 48105

1. W. Kool, F. A. Cushman, S. J. Gershman, When does model-based control pay off? *PLoS Comput. Biol.* **12**, e1005090 (2016).
2. W. Carvalho, M. S. Tomov, W. de Cothi, C. Barry, S. J. Gershman, Predictive representations: Building blocks of intelligence. *Neural Comput.* **36**, 1–74 (2024).
3. V. Mnih *et al.*, Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
4. R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming" in *Machine Learning Proceedings 1990* (Elsevier, 1990), pp. 216–224.
5. S. J. Gershman, A. B. Markman, A. R. Otto, Retrospective revaluation in sequential decision making: A tale of two systems. *J. Exp. Psychol. Gen.* **143**, 182 (2014).
6. T. Schaul, D. Horgan, K. Gregor, D. Silver, "Universal value function approximators" in *International Conference on Machine Learning* (PMLR, 2015), pp. 1312–1320.
7. L. P. Kaelbling, "Learning to achieve goals" in *IJCAI* (Citeseer, 1993), vol. 2, pp. 1094–1098.
8. M. S. Tomov, E. Schulz, S. J. Gershman, Multi-task reinforcement learning in humans. *Nat. Hum. Behav.* **5**, 764–773 (2021).
9. OEL Team, et al., Open-ended learning leads to generally capable agents. arXiv [Preprint] (2021). http://arxiv.org/abs/2107.12808 (Accessed 4 November 2026).
10. A. Barreto *et al.*, Successor features for transfer in reinforcement learning. *Adv. Neural Inf. Process. Syst.* **30**, 05312 (2017).
11. A. Barreto *et al.*, "Transfer in deep reinforcement learning using successor features and generalised policy improvement" in *International Conference on Machine Learning* (PMLR, 2018). pp. 501–510.
12. I. Momennejad *et al.*, The successor representation in human reinforcement learning. *Nat. Hum. Behav.* **1**, 680–692 (2017).
13. I. Momennejad, A. R. Otto, N. D. Daw, K. A. Norman, Offline replay supports planning in human reinforcement learning. *eLife* **7**, e32548 (2018).

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.

14. W. E. Hick, On the rate of gain of information. *Q. J. Exp. Psychol.* **4**, 11–26 (1952).
15. F. C. Donders, On the speed of mental processes. *Acta Psychol.* **30**, 412–431 (1969).
16. S. Sternberg, Memory-scanning: Mental processes revealed by reaction-time experiments. *Am. Scientist* **57**, 421–457 (1969).
17. Q. J. Huys *et al*., Interplay of approximate planning strategies. *Proc. Natl. Acad. Sci. U.S.A.* **112**, 3098–3103 (2015).
18. M. Matthews *et al*., "Craftax: A lightning-fast benchmark for open-ended reinforcement learning" in *International Conference on Machine Learning (ICML)* (2024).
19. R. Agarwal, M. C. Machado, P. S. Castro, M. G. Bellemare, Contrastive behavioral similarity embeddings for generalization in reinforcement learning. arXiv [Preprint] (2021). http://arxiv.org/abs/2101.05265 (Accessed 4 November 2026).
20. T. D. Kulkarni, K. Narasimhan, A. Saeedi, J. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Adv. Neural Inform. Process. Syst.* **29**, 06057 (2016).
21. D. Borsa, et al., Universal successor features approximators. arXiv [Preprint] (2018). http://arxiv.org/abs/1812.07626 (Accessed 4 November 2026).
22. T. Gerstenberg, M. F. Peterson, N. D. Goodman, D. A. Lagnado, J. B. Tenenbaum, Eye-tracking causality. *Psychol. Sci.* **28**, 1731–1744 (2017).
23. A. Norbury, T. W. Robbins, B. Seymour, Value generalization in human avoidance learning. *eLife* **7**, e34779 (2018).
24. E. Schulz, E. Konstantinidis, M. Speekenbrink, Putting bandits into context: How function learning supports decision making. *J. Exp. Psychol.: Learn., Memory, Cogn.* **44**, 927 (2018).
25. C. M. Wu, B. Meder, E. Schulz, Unifying principles of generalization: Past, present, and future. *Annu. Rev. Psychol.* **76**, 275–302 (2024).
26. S. Hall-McMaster, M. S. Tomov, S. J. Gershman, N. W. Schuck, Neural evidence that humans reuse strategies to solve new tasks. *PLoS Biol.* **23**, e3003174 (2025).
27. M. Andrychowicz *et al*., Hindsight experience replay. *Adv. Neural Inform. Process. Syst.* **30**, 01495 (2017).
28. R Yang, et al., Mher: Model-based hindsight experience replay. arXiv [Preprint] (2021). http://arxiv.org/abs/2107.00306 (Accessed 4 November 2026).
29. Z. D. Guo, E. Brunskill, Directed exploration for reinforcement learning. arXiv [Preprint] (2019). http://arxiv.org/abs/1906.07805 (Accessed 4 November 2026).
30. Z. Ren, K. Dong, Y. Zhou, Q. Liu, J. Peng, Exploration via hindsight goal generation. *Adv. Neural Inf. Process. Syst.* **32**, 04279 (2019).
31. L. Moro, A. Likmeta, E. Prati, M. Restelli, et al., "Goal-directed planning via hindsight experience replay" in *10th International Conference on Learning Representations, ICLR 2022* (2022). pp. 1–16.
32. V. Veeriah, J. Oh, S. Singh, Many-goals reinforcement learning. arXiv [Preprint] (2018). http://arxiv.org/abs/1806.09605 (Accessed 4 November 2026).
33. R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, D. Pathak, Discovering and achieving goals via world models. *Adv. Neural Inf. Process. Syst.* **34**, 24379–24391 (2021).
34. A. Brohan, et al., Rt-2: Vision-language-action models transfer web knowledge to robotic control. arXiv [Preprint] (2023). http://arxiv.org/abs/2307.15818 (Accessed 4 November 2026).
35. J. Schrittwieser *et al*., Mastering atari, go, chess and shogi by planning with a learned model. *Nature* **588**, 604–609 (2020).
36. M. Savva, et al., "Habitat: A platform for embodied ai research" in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 9339–9347.
37. A. Szot *et al*., Habitat 2.0: Training home assistants to rearrange their habitat. *Adv. Neural Inform. Process. Syst.* **34**, 251–266 (2021).
38. W. Carvalho, et al., Reinforcement learning for sparse-reward object-interaction tasks in a first-person simulated 3d environment. arXiv [Preprint] (2020). http://arxiv.org/abs/2010.15195 (Accessed 4 November 2026).
39. G. Dragoi, S. Tonegawa, Preplay of future place cell sequences by hippocampal cellular assemblies. *Nature* **469**, 397–401 (2011).
40. G. Dragoi, S. Tonegawa, Distinct preplay of multiple novel spatial experiences in the rat. *Proc. Natl. Acad. Sci. U.S.A.* **110**, 9100–9105 (2013).
41. H. F. Ólafsdóttir, C. Barry, A. B. Saleem, D. Hassabis, H. J. Spiers, Hippocampal place cells construct reward related sequences through unexplored space. *eLife* **4**, e06063. (2015).
42. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2018).
43. J. Peng, R. J. Williams, Incremental multi-step Q-learning. *Mach. Learn.* **22**, 283–290 (1996).
44. A. Kumar, A. Zhou, G. Tucker, S. Levine, "Conservative Q-learning for offline reinforcement learning" in *Advances in Neural Information Processing Systems* (2020), vol. 33, pp. 1179–1191.
45. B. Kim, Mh Oh, "Peng's Q(*lambda*) for conservative value estimation in offline reinforcement learning" in *The Fourteenth International Conference on Learning Representations* (2026).
46. W. Carvalho, Multi-task preplay in humans and machines. Open Science Framework. https://doi.org/10.17605/OSF.IO/M53QH. Accessed 4 November 2026.
47. W. Carvalho, Multi-task preplay in humans and machines - Crafter. Open Science Framework. https://doi.org/10.17605/OSF.IO/B2EVM. Accessed 4 November 2026.
48. W. Carvalho, multitask_preplay. GitHub. https://github.com/wcarvalho/multitask_preplay. Accessed 4 November 2026.

Downloaded from https://www.pnas.org by HARVARD UNIVERSITY; CABOT SCIENCE LIBRARY on November 19, 2026 from IP address 65.112.8.196.