When to Plan: Learning to Select Between Reactive Control and Deliberative Planning

Adam Labiosa, Josiah P. Hanna

Keywords: Meta-reasoning, reinforcement learning, adaptive computation.

Summary

It has long been recognized that humans have the ability to switch between fast, reactive decision-making and slower, deliberative planning. In this paper, we study the question of how to learn this ability, known as meta-reasoning, in artificial agents. We model reactive decision-making as a policy that directly maps state observations to actions. Such policies can be trained with reinforcement learning (RL) or imitation learning, but may generalize poorly outside of their training distribution. Alternatively, model-based decision-time planning is more likely to produce good actions across a broader set of states but requires additional computation time, which delays acting. In this work, we introduce an RL method for training a meta-reasoning policy that allocates computation by conditioning on a reactive-policy uncertainty score. This score enables it to predict when the reactive policy is likely to perform poorly and when planning is needed. We conduct an empirical study on motion planning and navigation environments, showing that this design enables the meta-reasoning policy to learn when the reactive policy provides a good-enough action versus when decision-time planning is needed. Additionally, we show that our design enables the meta-agent to shift toward fully reactive control as the reactive policy improves.

Contribution(s)

1. We introduce a reinforcement learning approach to learning a meta-policy for selecting between a fast, reactive policy, which may perform poorly out of its training distribution, and a planner, which requires time but reliably produces near-optimal actions.
   Context: Prior work has used RL to minimize compute, either for a single decision by selecting internal computations (Callaway et al., 2018) or across an episode by using action repetitions (Orenstein et al., 2025). Our work differs by addressing the meta-reasoning problem as needing to make a sequence of meta-level selections between a reactive policy and a planner across a full task episode. Related ideas have been studied in cognitive science (Keramati et al., 2011; Kool et al., 2018), but these works aim to describe human behavior rather than provide mechanisms for artificial agents.
2. We show that our method produces meta-policies which balance deliberate planning and using a reactive policy as a function of environment characteristics such as planning delay, reactive policy competence, and environment stochasticity.
   Context: We conduct an empirical study which shows our method obtains higher return than baselines that always commit to one type of decision-rule (Table 1). By varying planning cost, reactive policy competence, and task stochasticity, we analyze how each factor changes the balance of reactive action compared to deliberative planning (Figure 3).
3. We show that an uncertainty-conditioned meta-policy can track reactive policy competence even if the reactive policy improves over time, shifting toward more reactive control as reactive competence increases.
   Context: We conduct an experiment which uses behavior cloning on planner rollouts to fine-tune the reactive policy during training (Section 6.6, Figure 4). The key result is that uncertainty-conditioned observations are sufficient for the meta-policy to learn to call the reactive policy more as its training distribution widens.

When to Plan: Learning to Select Between Reactive Control and Deliberative Planning

Adam Labiosa[1], **Josiah P. Hanna**[1]
labiosa@wisc.edu, jphanna@cs.wisc.edu

[1]**Department of Computer Sciences, University of Wisconsin–Madison, USA**

Abstract

It has long been recognized that humans have the ability to switch between fast, reactive decision-making and slower, deliberative planning. In this paper, we study the question of how to learn this ability, known as meta-reasoning, in artificial agents. We model reactive decision-making as a policy that directly maps state observations to actions. Such policies can be trained with reinforcement learning (RL) or imitation learning, but may generalize poorly outside of their training distribution. Alternatively, model-based decision-time planning is more likely to produce good actions across a broader set of states but requires additional computation time, which delays acting. In this work, we introduce an RL method for training a meta-reasoning policy that allocates computation by conditioning on a reactive-policy uncertainty score. This score enables it to predict when the reactive policy is likely to perform poorly and when planning is needed. We conduct an empirical study on motion planning and navigation environments, showing that this design enables the meta-reasoning policy to learn when the reactive policy provides a good-enough action versus when decision-time planning is needed. Additionally, we show that our design enables the meta-agent to shift toward fully reactive control as the reactive policy improves.

1 Introduction

Humans naturally allocate cognitive effort depending on the situation. For example, drivers often navigate familiar routes with minimal cognitive effort, whereas in unfamiliar environments they shift from following intuitive reactions to deliberate decision-making (Charlton & Starkey, 2013). Acting reactively as opposed to thinking before acting has been termed System 1 and System 2 thinking in cognitive science (Kahneman, 2011; Evans, 1984) and a similar distinction can be seen in AI systems. Reactive, pattern-matching agents trained with model-free RL or imitation learning enable quick decision-making but may generalize poorly to states outside of their training distribution whereas agents that always invoke a decision-time planner produce high-quality actions but must spend time creating a plan before acting. Most current systems commit to a fixed strategy at test time like, for example, always using model-free reactive actions (Bansal et al., 2018; Haarnoja et al., 2023), always planning before acting (Xia & Chen, 2024), or always reasoning before responding (OpenAI et al., 2024; DeepSeek-AI et al., 2025).

The goal of this work is to enable *adaptive computation* by learning *when* to invoke more computationally-intensive planning routines as opposed to taking a reactive action. Our work contributes to the area of AI research known as meta-reasoning (Horvitz, 1990; Russell & Wefald, 1991), the study of how an agent should allocate its computation, and specifically the use of RL to learn policies for meta-cognition.

We study the meta-reasoning problem in a setting with two primary action-generation mechanisms: a reactive policy and a decision-time planner. The reactive policy produces an action from a single

forward pass of a neural network but only performs well within a limited training distribution. In contrast, the planner produces a good sequence of actions from any environment state albeit at the cost of additional computation compared to the reactive policy. While learned world models used for planning are similarly limited to their training distribution, prior work has shown that they generalize more robustly than reactive policies (Young et al., 2023). Planning is therefore most valuable in states outside of the reactive policy's training distribution. These observations motivate the need to arbitrate between a distribution-constrained reactive policy and a planner that is effective across the full state space.

Within this problem setting, we ask the following question:

How can an RL agent learn to select between a distribution-constrained, reactive policy and a slower decision-time planning routine?

To answer this question, we develop a method to train a meta-policy that selects between reactive action and variable-depth planning to maximize task return under a time-based cost. Critically, the meta-policy is conditioned on uncertainty signals from the reactive policy which allow it to estimate reactive competence in the current state.

We evaluate our approach in a suite of motion planning and navigation tasks. First, we show our adaptive meta-policy takes less time to reach goal states in comparison to fixed-compute baselines including always react and always plan. Second, we ablate key design choices of the meta-policy observation space and show that reactive uncertainty and observation history are the most critical components for effective compute allocation. Third, we demonstrate the meta-policy learns environment characteristic-dependent behavior by varying the reactive competence, planning cost and environment stochasticity. Lastly, we apply our method to a joint-training setting where the reactive policy improves alongside the meta-policy and show that uncertainty-conditioned observations enable the meta-policy to track increasing reactive competence and shift toward fully reactive control.

In summary, our work makes the following contributions:

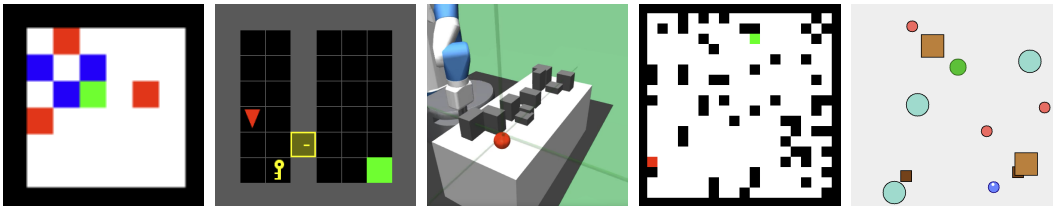
1. We introduce a novel reinforcement learning approach to learning a meta-policy for selecting between a fast, reactive policy, which may perform poorly out of its training distribution, and a planner, which requires time but reliably produces near-optimal actions.
2. We show that our method produces meta-policies which balance deliberate planning and using a reactive policy as a function of environment characteristics such as planning delay, reactive policy competence, and environment stochasticity.
3. We show that an uncertainty-conditioned meta-policy can track reactive policy competence even if the reactive policy improves over time, shifting toward more reactive control as reactive competence increases.

2 Markov Decision Processes

We consider a Markov decision process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ denotes the transition dynamics, $r(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each time step t, the agent selects an action $a_t \sim \pi(\cdot \mid s_t)$ according to a policy π, inducing a trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$. The RL objective is to learn a policy that maximizes the expected discounted return: $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.

3 Related Work

Learning to use computation. Several works use RL to learn meta-reasoning. Similar to our work, Callaway et al. (2018) uses model-free RL to select computations. However, their work focuses on allocating compute before taking an action in the real world while our work frames



(a) Box Push (b) Doorkey (c) Fetch (d) Maze (e) Navigation

Figure 1: Example images of our environment suite. Environments vary in state space type (discrete versus continuous), task structure, and reactive policy difficulty. In all environments the objective is to reach a goal state from a start state.

computation allocation as a sequential decision problem that persists across a full task episode. Other works adapt the amount of compute within a planner (Budd et al., 2024; Wang et al., 2025), whereas our agent decides between executing a reactive policy instantly or spending time to generate a plan. Event-triggered and adaptive replanning methods (Hashimoto et al., 2025; Chen et al., 2022; Honda et al., 2024) learn when a plan is no longer valid, but assume an agent that always plans and does not consider minimizing computation to achieve faster task completion. Other related works consider variable-length options to minimize the number of actions (Karimi et al., 2023; Lakshminarayanan et al., 2017a; Metelli et al., 2020; Orenstein et al., 2025; Sharma et al., 2020) but these works optimize action efficiency rather than computation allocation, and do not consider an explicit tradeoff between reactive execution and planning cost.

Arbitration between model-free and model-based control. The question of when to use model-free versus model-based control has been studied in both AI and neuroscience (Dollé et al., 2010; Lee et al., 2014). Existing AI approaches typically rely on hand-designed switching rules or uncertainty heuristics derived from prediction error signals (Sheikhnezhad Fard & Trappenberg, 2019; Fard & Trappenberg, 2018; Sharma et al., 2025; Raj et al., 2024), and ignore the computation required for model-based planning. Other methods learn to select among multiple controllers (Kästner et al., 2021; Dey et al., 2023; Hamrick et al., 2017) but assume planning and reactive inference take equal time. Our work differs by explicitly modeling the time-based cost of different controllers and learning a meta-reasoning policy to arbitrate between them.

Adaptive computation in LLMs. Recent work on LLMs addresses when to generate reasoning tokens (Paglieri et al., 2025; Fang et al., 2025; Sabbata et al., 2025; Manvi et al., 2025; Chung et al., 2025). While structurally similar, these works do not use meta-level agents and instead aim to encode the decision into the LLMs themselves, which constrains the adaptability of the methods. Ong et al. (2025) uses a controller to route between a larger and smaller model but is trained on preference data instead of learning the decision online as ours does. Hanna & Corrado (2025) introduce a theoretical formalism for when an RL agent should learn to take abstract thinking actions that can be interpreted as choosing additional computation. In contrast, we focus on developing agents that choose from among a set of concrete decision-rules that use varying amounts of computation.

Cognitive science. The tradeoff between fast reactive behavior and slow deliberation is well-studied in cognitive science under bounded rationality and dual-process theories (Griffiths et al., 2019; Keramati et al., 2011; Kelly & Barron, 2022). Several works theorize that humans deliberate when the expected value exceeds its cost (Lieder & Griffiths, 2017; Lieder et al., 2018), which directly motivates our formulation of planning as a time-costly option. Similar work applies RL as a model of meta-cognitive control (Krueger et al., 2017; Jain et al., 2019), but these works provide descriptive models of human behavior whereas we focus on operationalizing this ability in artificial agents.

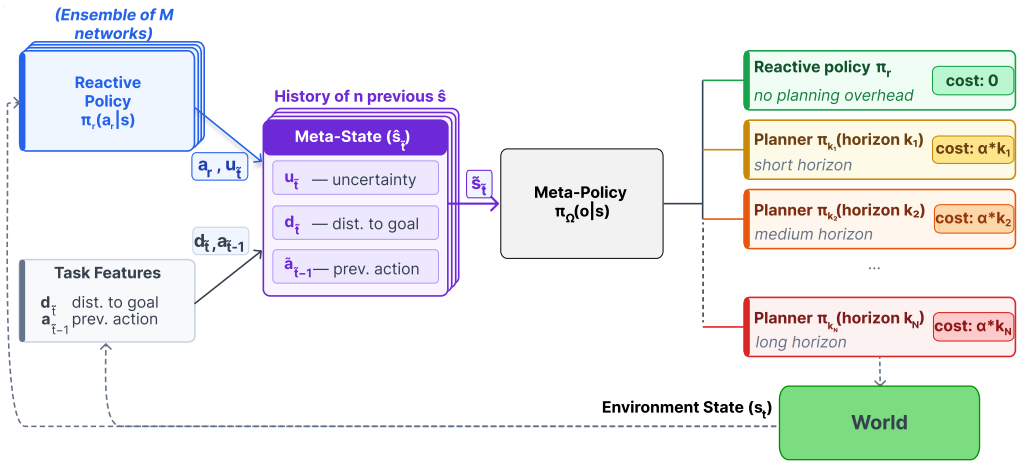


Figure 2: The meta reasoning agent. The reactive policy (ensemble of neural networks), outputs an action a_r and an uncertainty score $u_{\tilde{t}}$ for the current state. a_r and $u_{\tilde{t}}$ along with task features of distance to goal and previous action are input to the meta-agent. The meta-agent chooses between a reactive action and a set of plans with different horizons. The actions are executed in the environment.

4 Problem Formulation

We consider the problem of training, with RL, a meta-reasoning policy that selects between controllers with different time-based costs. Specifically, we are interested in switching between a reactive policy, which produces correct actions within its training distribution but costs little time, and a planner, which produces correct actions across the entire state space but costs more time. We formalize this meta-reasoning problem using the options framework (Sutton et al., 1999). The meta-MDP is defined as $\tilde{\mathcal{M}} = (\tilde{\mathcal{S}}, \mathcal{O}, \tilde{\mathcal{P}}, \tilde{r}, \gamma)$ where $\tilde{\mathcal{S}}$ is the meta-state space, $\mathcal{O}$ is the options space, where an option $o \in \mathcal{O}$ is a possibly temporally-extended choice action that executes in the world environment, $\tilde{\mathcal{P}}$ is the meta-transition function, $\tilde{r}$ is the meta-reward and γ is the discount factor.

The meta-state space $\tilde{\mathcal{S}}$ is the space of the agent's possible internal states, which need not correspond directly to the world-environment state space $\mathcal{S}$. There are two types of options $\mathcal{O}$ available to the meta-agent: planning options and reactive control options. We are particularly interested in understanding how the tradeoff between decision speed and decision quality affects task performance. With this in mind, we restrict ourselves to goal-reaching domains where optimal behavior minimizes time to reach the goal state.

Planning options correspond to selecting a planner π_k constrained to a fixed planning horizon k. For a plan option of horizon k, the planner computes a sequence $p = [a_0, a_1, \ldots, a_{k-1}]$ which is then executed to completion in the world environment. We unify the cost to plan and cost to act under a time-based scale to enable optimizing under the time-to-goal objective. We define the plan option cost as $c_p(k) = -\alpha k$ where α scales the time-cost per search depth. Each execution of an action incurs a cost $c_a = -1$ (i.e. takes one unit of time), so the total reward of executing a planning option is $\tilde{r}_{\tilde{t}} = c_p(k) + k \cdot c_a = -\alpha k - k$.

Reactive control is modeled as a one-step option where the policy is a learned reactive policy $\pi_r(a \mid s)$. This option incurs no planning cost ($c_p = 0$) and executes a single environment action, therefore $\tilde{r}_{\tilde{t}} = c_a = -1$. The reactive policy is trained, through imitation learning, so that it produces reasonable actions on a limited amount of the full state space. Formally, the reactive policy is trained on a subset of states $\mathcal{S}_{ID} \subset \mathcal{S}$, with the remaining states $\mathcal{S}_{OOD} = \mathcal{S}/\mathcal{S}_{ID}$ considered out-of-distribution. For in-distribution states, the reactive policy performs near-optimally such that $V^{\pi_r}(s) \approx V^*(s)$ for $s \in \mathcal{S}_{ID}$. For out-of-distribution states $s \in \mathcal{S}_{OOD}$, the reactive policy may

generalize poorly such that $V^{\pi_r}(s) \ll V^*(s)$, motivating the use of planning despite the incurred computational cost.

In the meta-transition dynamics $\tilde{\mathcal{P}}$, transitions between states are induced by the underlying world-dynamics and the meta-action chosen. When a planning option of horizon k is taken, the meta-state transition occurs after k world-environment steps. When a reactive control option is chosen, the transition occurs after one world-environment step. The only time a k step planning option terminates early is when the underlying world-environment terminates partially through an option. At each meta-decision timestep $\tilde{t}$, the meta-policy $\pi_\Omega(o \mid \tilde{s}_{\tilde{t}})$ selects an option o based on the agent's internal information state $\tilde{s}_{\tilde{t}}$ to control the agent's use of computation. When an option terminates, the meta-policy observes the updated internal state and selects the next option.

5 Reinforcement Learning-Based Meta-Agent for When to Plan

In this section, we describe how to learn a meta-policy for allocating computation between reactive control and decision-time planning. Our meta-observation design is motivated by three properties of agents operating in large, open-ended environments. First, as the reactive policy will encounter both in-distribution and out-of-distribution states, the meta-observation cannot use raw environment state, since the meta-policy would then face the same in-distribution/out-of-distribution problem. Second, the reactive policy may improve over time through experience (Anthony et al., 2017), so the meta-observation must reflect current reactive competence. Third, since the primary benefit of the reactive policy is that it requires no planning time, the meta-observation must itself be computable without search.

With these observations in mind, we instantiate the meta-MDP as follows. The meta-policy operates on an internal state $\tilde{s}$ which can be computed without search. The observation consists of an uncertainty score from the reactive policy $u_{\tilde{t}}$, the current distance to the goal $d_{\tilde{t}}$, and the previous meta-action taken $\tilde{a}_{\tilde{t}-1}$. These values are concatenated as a single observation:

$$\hat{s}_{\tilde{t}} = [u_{\tilde{t}}, d_{\tilde{t}}, \tilde{a}_{\tilde{t}-1}]$$

Additionally, we provide the meta-agent with a history of n meta-observations $\hat{s}_{\tilde{t}}$, giving it access to trends in uncertainty and progress toward the goal:

$$\tilde{s}_{\tilde{t}} = [\hat{s}_{\tilde{t}}, \hat{s}_{\tilde{t}-1}, \ldots, \hat{s}_{\tilde{t}-(n-1)}]$$

The action space for our meta-agent includes a reactive option and N planning policy options:

$$\mathcal{O} = \{\pi_r, \pi_{k_1}, \pi_{k_2}, \ldots, \pi_{k_N}\}$$

Each planning option corresponds to a fixed planning horizon, with longer horizons producing longer plans at a greater time cost.

A meta-decision is made whenever an option terminates. At each meta-step (Figure 2), the reactive policy produces an action and uncertainty value, the meta-state is constructed, and the meta-policy selects an option. If the reactive option π_r is chosen, the reactive policy's action is executed for one world step. If a planning option π_k is chosen, a plan of horizon k is computed using the planner and then executed open-loop in the world environment for its full duration. The meta-policy π_Ω is trained with model-free RL to minimize time to the goal.

To compute the uncertainty score of the reactive policy for the meta-state, we implement our reactive policy as an ensemble of M neural networks (Lakshminarayanan et al., 2017b). The networks directly map the environment state s to an action a, and using the logits of the output, we compute the uncertainty score for the meta-state. This uncertainty score is a task-agnostic estimate of in-distribution competence based on ensemble disagreement. For continuous control, each network

outputs an action mean $\mu_i \in \mathbb{R}^D$. The ensemble action and uncertainty are computed as:

$$a = \frac{1}{M}\sum_{i=1}^{M}\mu_i, \qquad u(s) = \sum_{j=1}^{D}\mathrm{Var}_i[\mu_{i,j}]\,.$$

For discrete action spaces, each network outputs a categorical distribution $\mathbf{p}_i$ and the averaged distribution is $\bar{\mathbf{P}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{p}_i$. The ensemble action and uncertainty are:

$$a = \arg\max_j \bar{P}(j), \qquad u(s) = -\sum_{j=1}^{J}\bar{P}(j)\log\bar{P}(j).$$

In both cases, $u_{\tilde{t}} = u(s_{\tilde{t}})$, and high uncertainty corresponds to out-of-distribution inputs where planning may be beneficial, as the reactive policy is more likely to produce worse actions and therefore take more time to reach the goal.

6 Experiments

We evaluate our method on a suite of environments in which the goal is to navigate from a starting location to a goal location (Figure 1). First, we show that our meta-policy reaches goal states in fewer timesteps than fixed-compute baselines including always-reactive and always-planning policies. Second, we ablate the design of the meta-agent's observation space, finding that reactive uncertainty and observation history are the most critical components. Third, we characterize how environment factors including planning cost and reactive competence change the learned allocation strategy. Finally, we show the meta-policy tracks improving reactive competence during joint training, shifting to fully reactive control as both the meta-policy and reactive policy converge.

6.1 Environments

Our environment suite (Figure 1) has five environments with varying characteristics. For each environment, we generate distinct sets of object configurations along with a start and goal location which we consider to be a task. Unless otherwise stated, we use 20 tasks and assign half to reactive in-distribution and half to reactive out-of-distribution. For all environments except Maze, each task has a single start and goal location. For Maze, each task has 5 start and goal configurations.

Here we provide a summary of each environment:

1. **Box Push**: A simplified version of Sokoban, a standard planning benchmark where the goal is to push boxes onto goal locations (Botea et al., 2003). In our implementation the available gridspace is 6x6 and there are three boxes. In this domain, an incorrect move (e.g., moving a box into an incorrect corner) can be irrecoverable and cause task failure. The reactive actors receive a symbolic representation of the world.
2. **DoorKey**: An environment from the MiniGrid library (Chevalier-Boisvert et al., 2023). DoorKey is a discrete, multi-stage task where the objective is to navigate to a goal position by picking up a key, unlocking a door and walking to the goal. We modified the environment to use symbolic observations instead of the original image observations.
3. **Fetch**: An environment modified from the Gymnasium Robotics Fetch Reach environment (de Lazcano et al., 2024). We added in obstacles to the table which the gripper must navigate around. The reactive actors receive two images of the state of the world from different angles. The planner for this domain operates in Cartesian space while the reactive actor operates in 7 DOF joint space to make planner easier while preserving the difficulty of multi-joint arm control. We convert from Cartesian space to joint space through MuJoCo inverse kinematic control for reactive policy training.
4. **Maze**: A maze-like gridworld environment where the goal is to navigate to a goal location. Each obstacle configuration defines 5 start/goal pairs to test whether the meta-policy generalizes across targets within the same layout. The reactive actors receive the full maze as an image input.

Environment	Meta-policy	Reactive	Short	Medium	Long
Box Push	$\mathbf{-24.3}$ [0.4, 0.4]	-133.0 [3.1, 3.1]	-306.1 [3.6, 3.5]	-44.7 [2.0, 1.9]	-59.8 [3.7, 3.6]
Doorkey	$\mathbf{-25.1}$ [0.3, 0.3]	-135.2 [2.5, 2.4]	-126.7 [2.9, 2.9]	-27.2 [0.1, 0.1]	-32.0 [0.1, 0.1]
Fetch	$\mathbf{-12.2}$ [0.1, 0.1]	-132.4 [2.4, 2.4]	-51.0 [2.5, 2.4]	-70.7 [3.9, 3.7]	-30.0 [0.3, 0.3]
Maze	$\mathbf{-19.3}$ [0.3, 0.3]	-56.2 [0.9, 0.9]	-89.6 [1.4, 1.4]	-37.8 [1.0, 1.0]	-24.5 [0.4, 0.4]
Navigation	$\mathbf{-39.6}$ [0.1, 0.1]	-140.8 [2.3, 2.3]	-41.2 [0.1, 0.1]	-45.4 [0.1, 0.1]	-46.9 [0.2, 0.2]

Table 1: Average episode return which includes both cost of acting and planning. Baselines are fixed policies which exclusively call the reactive, short plan, medium plan, and long plan option. 95% bootstrap confidence intervals (margins shown below the mean). Bold entries overlap in CI with the best performer per environment. We run 30 seeds per experiment with 300 evaluation episodes per seed.

5. **Navigation**: A continuous navigation task. The goal of this environment is to navigate from a start position to a goal position through continuous control. The agent navigates by controlling continuous displacement at each timestep up to a small maximum distance in all directions. The reactive actors receive the full environment as an image input.

6.2 Experimental Setup

Meta-Policy. We train the meta-policy with PPO (Schulman et al., 2017) implemented in Stable Baselines3 (Raffin et al., 2021) with default hyperparameters. We chose PPO for its stability and well-understood training dynamics. We use a history of 4 observations to capture the short-term trends in reactive uncertainty and task progress.

Reactive Policy. The reactive policy is pretrained by behavior cloning from trajectories generated by the planner on in-distribution tasks. All experiments use $M = 4$ networks. For pretraining parameters see Appendix 4.

Planning. Planning is performed using a given accurate world model which isolates the compute allocation problem from the model learning problem. In all environments we use an A* planner which we constrain to a maximum search depth to approximate a time limit[1] (Table 4).

For continuous control domains, we use a lattice-based A* over a discretized Euclidean state space — end-effector space for Fetch and agent location space for Navigation. We use short, medium, and long planning horizons of 15%, 30%, and 50% of the task horizon to span a low, medium and high amount of compute. We leave the sensitivity of the meta-policy to these values for future work.

We use a plan cost of $\alpha = 0.5$ as the default value. This value was chosen to avoid planning that is prohibitively expensive or trivially cheap. A value of $\alpha = 0.5$ intuitively means that each planning step costs 50% more time units than a reactive step and avoids the two non-interesting regimes. As $\alpha \to 0$, planning is effectively free so an optimal policy always plans and when α is large planning is dominated by reactive control in all but the most extreme out-of-distribution cases. We analyze the sensitivity of the meta-policy to α in Section 6.5.

[1]Several proxies for a time-based budget are possible. In this work, we assign cost based on search depth. An alternative is an effort-based approximation, such as assigning the cost to be the number of A* nodes expanded. Both are proxies for the same underlying time-based cost.

Environment	Reactive	Short	Medium	Long
Box Push	44.4 [0.8, 0.8]	0.0 [0.0, 0.0]	0.0 [0.0, 0.0]	55.6 [0.8, 0.8]
DoorKey	23.8 [0.6, 0.7]	35.4 [0.3, 0.3]	40.8 [0.4, 0.4]	0.0 [0.0, 0.0]
Fetch	77.0 [0.5, 0.5]	20.7 [0.5, 0.5]	0.0 [0.0, 0.0]	2.3 [0.2, 0.2]
Maze	51.1 [1.0, 1.0]	0.3 [0.1, 0.1]	0.7 [0.1, 0.1]	47.9 [1.0, 1.0]
Navigation	20.0 [0.7, 0.7]	80.0 [0.7, 0.7]	0.0 [0.0, 0.0]	0.0 [0.0, 0.0]

Table 2: Meta-action percentages for the trained meta-policy with 95% bootstrap confidence intervals (margins shown below each mean). We run 30 seeds per experiment with 300 evaluation episodes per seed.

Evaluation. We define return to be the total cost of time to plan and act over an episode as stated in Section 4. We run all experiments over 30 independent seeds and results are aggregated across seeds. All methods are evaluated for 300 episodes using deterministic evaluation after training and we fix the hyperparameters across all ablations.

6.3 Learning When to Plan

In this experiment, we verify that our meta-policy learns to effectively allocate compute across our suite of environments. Table 1 shows the mean evaluation return for our meta-agent in comparison to fixed-compute baselines and Table 2 shows the meta-action distribution of the learned meta-policy. In all five domains, our meta-agent performs better than all fixed compute strategies, showing our adaptive computation method is able to learn effective planning-reacting behavior. The closest heuristic to our meta-agent is the Medium Plan in the Doorkey environment. Notably, Doorkey is also the only domain where the meta-policy does not converge to a two-action strategy. Since Doorkey is a staged task, these results suggest that the medium and short planning horizons align with the sub-goal structure of the task and a fixed strategy can achieve close performance to an adaptive one.

The two tasks where the meta-agent achieves the largest percent improvement over baselines are Box Push with almost 2x improvement and Fetch with about a 2.5x improvement. In Box Push, short-horizon planning is suboptimal in most situations since low-depth search is inadequate to find the long-horizon action sequences required to reach the goal state, as seen in the large performance gap between short and medium planning results. Furthermore, Box Push is the only domain where mistakes are unrecoverable, meaning imprecise use of the reactive policy risks failure. The meta-agent's performance suggests it learns to avoid reactive control and short plans in states where mistakes would cause failure. In Fetch, the complexity stems from the action space of the 7 DOF robot arm the reactive policy must control. Since the meta-agent converges to a predominately reactive strategy with over 75% reactive control, it suggests that the reactive policy partially generalizes to out-of-distribution tasks, but cannot complete them without planning. The strong performance in both domains suggests that task complexity does not make the meta-reasoning problem harder.

In three of the five tasks, the meta-policy converges to a near-equal split between reactive and planning actions, suggesting the agent learns to distinguish between in-distribution and out-of-distribution states. Navigation and Fetch are two exceptions, which likely reflect the reactive policy's out-of-distribution performance. Qualitatively, the navigation reactive policy sometimes gets

Environment	Meta-policy	Distance	History	Previous Action	Uncertainty
Box Push	-24.3 [0.4, 0.4]	$\mathbf{-20.9}$ [0.3, 0.2]	-110.7 [2.5, 2.5]	-36.3 [1.2, 1.1]	-73.9 [2.4, 2.2]
DoorKey	$\mathbf{-25.1}$ [0.3, 0.3]	-34.3 [1.0, 0.9]	-26.1 [0.1, 0.1]	-29.1 [0.6, 0.6]	-27.9 [0.6, 0.6]
Fetch	$\mathbf{-12.2}$ [0.1, 0.1]	-12.5 [0.1, 0.1]	-21.1 [0.8, 0.8]	$\mathbf{-12.1}$ [0.1, 0.1]	-16.1 [0.2, 0.2]
Maze	$\mathbf{-19.3}$ [0.4, 0.3]	$\mathbf{-19.7}$ [0.4, 0.4]	$\mathbf{-19.1}$ [0.3, 0.3]	$\mathbf{-19.3}$ [0.4, 0.4]	-25.0 [0.4, 0.4]
Navigation	-39.6 [0.1, 0.1]	-41.2 [0.1, 0.1]	-40.7 [0.1, 0.1]	$\mathbf{-37.9}$ [0.2, 0.2]	-41.2 [0.1, 0.1]
Average	$\mathbf{-24.1}$ [0.1, 0.1]	-25.7 [0.2, 0.2]	-43.5 [0.5, 0.5]	-27.0 [0.3, 0.3]	-36.8 [0.5, 0.5]

Table 3: Average episode return per method ablation with 95% bootstrap confidence intervals (margins below each mean). Bold entries overlap in CI with the best performer per environment. Each header is the component of the ablation we remove from the observation space (e.g., **Distance** removes the distance to goal). We run 30 seeds per experiment with 300 evaluation episodes per seed.

"stuck" on objects in the task, and the results suggest that the meta-policy can distinguish when that happened and switch to planning.

6.4 Ablation Study

In this section we investigate which components of our meta-agent's observation space are most critical for learning adaptive computation. We ablate each component by removing one feature at a time:

1. **Distance**: Removes the distance component.
2. **History**: Meta-agent only receives current state observations.
3. **Previous Action**: Removes the previous action taken.
4. **Uncertainty**: Removes the model-free uncertainty.

The results in Table 3 show that the two most important components to our method are the history and uncertainty scores. Poor performance without History is most clearly seen in the Box Push (4.5x worse) and Fetch (1.7x worse) environments. We once again note that these tasks have challenges in irreversibility of actions and high-dimensional action space respectively. By removing history, it seems as if the agent is no longer able to track progress, and therefore it is unclear if the reactive agent is moving toward the goal. A similar drop-off is seen without the uncertainty score in these two environments. This suggests that without directly being able to tell if the task is in-distribution or out-of-distribution for the reactive policy, the meta-agent is not able to perform optimally. However, the lack of substantial drop-off without the uncertainty score in the other three environments suggests that the combination of history and distance enables the agent to learn a proxy for uncertainty in these environments, but that learned metric is not as effective as directly accessing the uncertainty.

6.5 Environment Characteristics

In this section we investigate how different environment characteristics affect the learned meta-policy behavior. First, we vary the proportion of in-distribution tasks by changing the number of out-of-distribution tasks. Results shown in Figure 3 in the leftmost section. As the proportion of in-distribution tasks increases, the meta-policy shifts toward reactive control, with the proportion

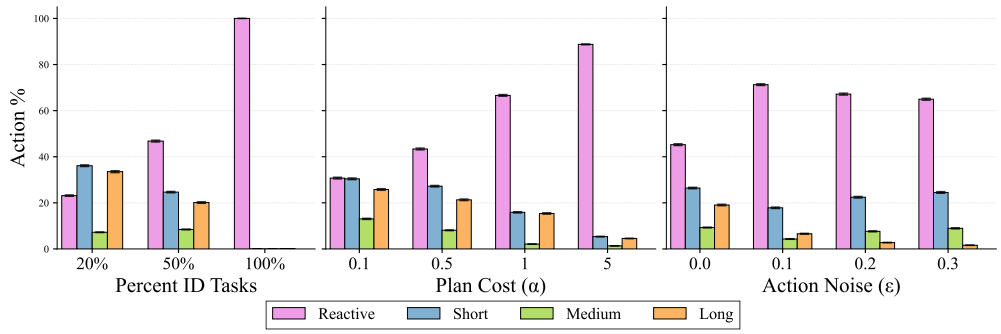


Figure 3: Average meta-action taken after training across all environments for different environment characteristics. We vary the percentage of in-distribution tasks (**left**), the computational cost of planning (**center**), and environmental stochasticity (**right**). Results reflect 95% bootstrap confidence intervals across 30 seeds per experiment (300 evaluation episodes per seed).

of reactive actions roughly tracking the proportion of in-distribution tasks. This behavior is expected since the reactive policy is competent in a higher proportion of tasks so the meta-policy correctly identifies that planning is less frequently necessary. At the extreme where all tasks are in-distribution, the meta-policy converges to fully reactive control, confirming the meta-policy correctly interprets the reactive policy's competence across the distribution of tasks.

Second, we vary the planning cost parameter α, where a value of $\alpha = 0.1$ means a plan of length 15 costs 1.5 timesteps to compute. Results are shown in Figure 3 in the middle section. As planning cost increases, the meta-policy relies increasingly on reactive control. Notably, even though the reactive policy produces suboptimal actions in out-of-distribution states, the time saved by avoiding planning is sufficient to achieve better returns than planning in some cases. A second finding is that the ratio of short, medium and long planning horizons remains roughly constant as α increases, suggesting the meta-policy reduces planning frequency uniformly rather than exclusively dropping longer plans. This implies the meta-policy learns when to plan, but not how long to plan, as the distribution of plan lengths remains consistent across environments.

Third, we vary the stochasticity of the environment by introducing action noise ϵ into the transition dynamics. For discrete environments, ϵ controls the probability of a random action being taken and for continuous environments, we add Gaussian noise with variance ϵ to the action. Results are shown in Figure 3 in the rightmost section. The most significant shift occurs between no noise and low noise ($\epsilon = 0.1$), where the reactive action percentage increases from approximately 45% to 70%. The increase in reactive action percentage is driven mostly by a reduction in long planning, which drops from roughly 20% to 5%. This result is consistent with the intuition that longer plans are most affected by increased noise since fewer replanning actions means errors compound more over the plan duration. As noise increases further, this trend continues with long planning decreasing with each increase in ϵ. Interestingly, as noise increases from $\epsilon = 0.1$ to $\epsilon = 0.3$, the reactive percentage decreases slightly while short and medium planning increase slightly. This result suggests that while the reactive policy handles high noise better than long planning, short and medium plans retain utility in out-of-distribution states since half of the tasks remain out-of-distribution for the reactive policy. The meta-policy therefore converges to a mixed strategy that favors reactive control more than the baseline $\epsilon = 0$ but sometimes uses short planning for out-of-distribution states under high stochasticity.

6.6 Joint-Training Setting

In this section we empirically evaluate the ability of our method to adapt in a joint-training setting where the reactive actor improves alongside the meta-agent. To train the reactive policy, we record the state-action pairs produced by planning options and add these to a replay buffer. After each

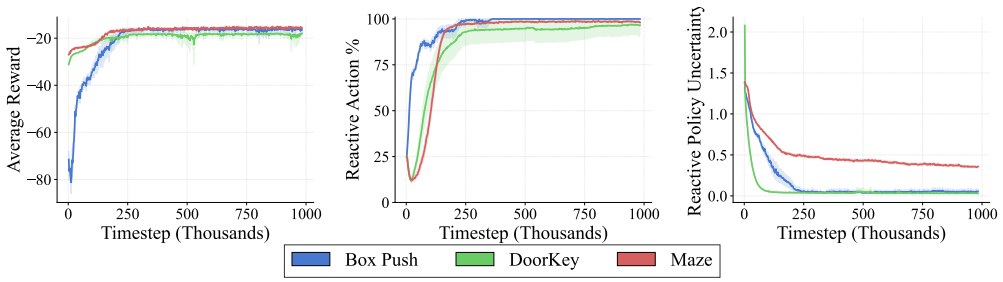


Figure 4: Average return, reactive action percent and reactive actor uncertainty value as the meta-policy is trained. The reactive policy is trained on a history buffer of planner actions after every rollout. 30 seeds per environment. Shaded regions indicate the 95% confidence interval. Box Push, DoorKey, and Maze only; Fetch and Navigation excluded due to compute constraints.

meta-agent rollout, all reactive ensemble members are trained by behavior cloning on this buffer. As joint training requires significantly longer training runs, we restrict this experiment to Box Push, DoorKey, and Maze due to compute constraints. Training hyperparameters are reported in Table 4.

Results are shown in Figure 4. As training progresses, the meta-policy shifts toward reactive control across all environments, converging to fully reactive execution as the reactive actor improves. This is also seen in the reactive uncertainty signal, which converges to low disagreement across environments and confirms the ensemble uncertainty score reflects the strength of the reactive policy. Together these results show the meta-policy is able to adapt as the reactive policy learns and converge to the optimal strategy.

An interesting property of this training setup is the reactive policy is trained on data generated from when a planning option is called. This structure creates a learning curriculum where the meta-agent must call planning in out-of-distribution states, which simultaneously produces the best immediate reward and generates the training data for the reactive policy. Once the reactive policy has been trained on out-of-distribution states, the meta-policy then recognizes the reactive policy uncertainty has decreased, revealing that the reactive policy is now competent in more states and stops calling planning options. This shift maximizes the return and avoids overtraining the reactive policy on already in-distribution states.

7 Discussion and Future Work

Our empirical results demonstrate that RL-based meta-policies can successfully learn adaptive computation allocation from uncertainty signals and high-level state features across a suite of motion planning and navigation tasks. However, several assumptions we make point toward natural future work. First, our method assumes access to a perfect world model for planning which does not hold in most real-world settings. Cognitive science work on meta-reasoning suggests that humans account for model uncertainty when deciding whether to plan (Gläscher et al., 2010) so extending our method to incorporate learned world models with state uncertainty is a natural next step.

Second, we constrain our environments to be single-agent and static during plan creation. Widening the environment suite to include tasks without such constraints and enabling the meta-agent to interrupt executing plans would test whether the reactive-planning tradeoff we study can generalize beyond the synchronous and single-agent setting.

Third, our meta-observation relies on hand-designed features. While our ablations confirm these are effective, more expressive uncertainty estimates (e.g., Bayesian or flow-based methods) or learned observation representations could improve meta-policy generalization, particularly in high-dimensional observation spaces such as images.

Our joint-training setup also suggests a promising direction for future work. Our meta-agent was able to track an improving reactive policy over the course of training so an interesting question is whether it can also track a *worsening* reactive policy. This is a setting that may occur when the reactive policy has insufficient capacity to represent optimal actions in all states across a big world (Javed & Sutton, 2024) or suffers from catastrophic forgetting.

More broadly, the reactive-planning tradeoff is applicable to many adaptive-computation problems. In LLM settings, the tradeoff between fast single-pass generation and slower chain-of-thought reasoning is structurally similar to the reactive-planning tradeoff we study in this work. Some VLA works (Gemini Robotics Team, 2025) adopt a similar structure with a fast and lightweight model on a robot and a larger but slower model in the cloud. Our uncertainty-conditioned meta-policy provides a method for learning these allocations.

8 Conclusion

In this work, we presented an RL method for learning adaptive computation allocation through a meta-reasoning policy. Our approach conditions on reactive policy uncertainty and high-level state features to select between a fast reactive option and planning options, where planning improves action quality at the cost of time. Across a suite of motion planning and navigation tasks, our meta-policy outperforms all fixed-compute heuristic baselines when optimizing for a time-to-goal objective. Ablations confirm that reactive uncertainty and observation history are the most critical components of the meta-state. We further showed that the learned meta-policy adapts to environment characteristics: increasing planning cost shifts allocation toward reactive control, and increasing reactive in-distribution coverage shifts allocation away from planning. Finally, we demonstrated that conditioning on uncertainty signals enables the meta-policy to converge to exclusively reactive control in a joint-training setting where the reactive policy continues to learn. Together, these results show that high-level state and reactive competence signals are sufficient for a meta-reasoning policy to effectively allocate computation across environments and training regimes.

Acknowledgments

We thank Brahma Pavse and the RLC reviewers for providing helpful feedback which improved this work before and during the review process. This work took place in the Prediction and Action Lab (PAL) at the University of Wisconsin–Madison. PAL research is supported by NSF (IIS-2410981) and the Wisconsin Alumni Research Foundation. Additionally, Adam Labiosa is supported by an NSF Research Traineeship award, No. 2152163.

References

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst, December 2018. URL http://arxiv.org/abs/1812.03079. arXiv:1812.03079 [cs].

Adi Botea, Martin Müller, and Jonathan Schaeffer. Using Abstraction for Planning in Sokoban. In Jonathan Schaeffer, Martin Müller, and Yngvi Björnsson (eds.), *Computers and Games*, pp. 360–375, Berlin, Heidelberg, 2003. Springer. ISBN 978-3-540-40031-8. DOI: 10.1007/978-3-540-40031-8_24.

Matthew Budd, Bruno Lacerda, and Nick Hawes. Stop! Planner Time: Metareasoning for Probabilistic Planning Using Learned Performance Profiles. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):20053–20060, March 2024. ISSN 2374-3468. DOI: 10.1609/aaai.v38i18.29983. URL https://ojs.aaai.org/index.php/AAAI/article/view/29983.

Frederick Callaway, Sayan Gul, Paul M. Krueger, Thomas L. Griffiths, and Falk Lieder. Learning to select computations, August 2018. URL http://arxiv.org/abs/1711.06892. arXiv:1711.06892 [cs].

Samuel G. Charlton and Nicola J. Starkey. Driving on familiar roads: Automaticity and inattention blindness. *Transportation Research Part F: Traffic Psychology and Behaviour*, 19:121–133, July 2013. ISSN 1369-8478. DOI: 10.1016/j.trf.2013.03.008. URL https://www.sciencedirect.com/science/article/pii/S1369847813000326.

Jun Chen, Xiangyu Meng, and Zhaojian Li. Reinforcement Learning-based Event-Triggered Model Predictive Control for Autonomous Vehicle Path Following. In *2022 American Control Conference (ACC)*, pp. 3342–3347, June 2022. DOI: 10.23919/ACC53348.2022.9867347. URL https://ieeexplore.ieee.org/document/9867347/. ISSN: 2378-5861.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Stephen Chung, Wenyu Du, and Jie Fu. Thinker: Learning to Think Fast and Slow, May 2025. URL http://arxiv.org/abs/2505.21097. arXiv:2505.21097 [cs].

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature*, 645(8081):633–638, September 2025. ISSN 0028-0836, 1476-4687. DOI: 10.1038/s41586-025-09422-z. URL http://arxiv.org/abs/2501.12948. arXiv:2501.12948 [cs].

Sombit Dey, Assem Sadek, Gianluca Monaci, Boris Chidlovskii, and Christian Wolf. Learning Whom to Trust in Navigation: Dynamically Switching Between Classical and Neural Planning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5235–5242, October 2023. DOI: 10.1109/IROS55552.2023.10342308. URL https://ieeexplore.ieee.org/document/10342308/. ISSN: 2153-0866.

Laurent Dollé, Denis Sheynikhovich, Benoît Girard, Ricardo Chavarriaga, and Agnès Guillot. Path planning versus cue responding: a bio-inspired model of switching between navigation strategies. *Biological Cybernetics*, 103(4):299–317, October 2010. ISSN 1432-0770. DOI: 10.1007/s00422-010-0400-z. URL https://doi.org/10.1007/s00422-010-0400-z.

Jonathan St BT Evans. Heuristic and analytic processes in reasoning. *British Journal of Psychology*, 75(4):451–468, 1984.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM Learns When to Think, June 2025. URL http://arxiv.org/abs/2505.13379. arXiv:2505.13379 [cs].

Farzaneh S. Fard and Thomas P. Trappenberg. Mixing Habits and Planning for Multi-Step Target Reaching Using Arbitrated Predictive Actor-Critic. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2018. DOI: 10.1109/IJCNN.2018.8489122. URL https://ieeexplore.ieee.org/document/8489122. ISSN: 2161-4407.

Gemini Robotics Team. Gemini Robotics: Bringing AI into the Physical World, March 2025. URL http://arxiv.org/abs/2503.20020. arXiv:2503.20020 [cs].

Jan Gläscher, Nathaniel Daw, Peter Dayan, and John P. O'Doherty. States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning. *Neuron*, 66(4):585–595, May 2010. ISSN 0896-6273. DOI: 10.1016/j.neuron.2010.04.016. URL https://www.cell.com/neuron/abstract/S0896-6273(10)00287-4.

Thomas L Griffiths, Frederick Callaway, Michael B Chang, Erin Grant, Paul M Krueger, and Falk Lieder. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29:24–30, October 2019. ISSN 23521546. DOI: 10.1016/j.cobeha.2019.01.005. URL https://linkinghub.elsevier.com/retrieve/pii/S2352154618302122.

Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever, Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess. Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning, April 2023. URL http://arxiv.org/abs/2304.13653. arXiv:2304.13653 [cs].

Jessica B. Hamrick, Andrew J. Ballard, Razvan Pascanu, Oriol Vinyals, Nicolas Heess, and Peter W. Battaglia. Metacontrol for Adaptive Imagination-Based Optimization, May 2017. URL https://arxiv.org/abs/1705.02670v1.

Josiah P. Hanna and Nicholas E. Corrado. When Can Model-Free Reinforcement Learning be Enough for Thinking?, October 2025. URL http://arxiv.org/abs/2506.17124. arXiv:2506.17124 [cs].

Kazumune Hashimoto, Yuga Onoue, Akifumi Wachi, and Xun Shen. Learning-Based Event-Triggered MPC With Gaussian Processes Under Terminal Constraints. *IEEE Transactions on Cybernetics*, 55(4):1512–1525, April 2025. ISSN 2168-2275. DOI: 10.1109/TCYB.2025.3536606. URL https://ieeexplore.ieee.org/document/10892326/.

Kohei Honda, Ryo Yonetani, Mai Nishimura, and Tadashi Kozuno. When to Replan? An Adaptive Replanning Strategy for Autonomous Navigation using Deep Reinforcement Learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6650–6656, May 2024. DOI: 10.1109/ICRA57147.2024.10611474. URL https://ieeexplore.ieee.org/document/10611474.

Eric Horvitz. Computation and action under bounded resources /. 1990.

Yash Raj Jain, Sanit Gupta, Vasundhara Rakesh, Peter Dayan, Frederick Callaway, and Falk Lieder. How do people learn how to plan? In *2019 Conference on Cognitive Computational Neuroscience*, Berlin, Germany, 2019. Cognitive Computational Neuroscience. DOI: 10.32470/CCN.2019.1313-0. URL https://ccneuro.org/2019/Papers/ViewPapers.asp?PaperNum=1313.

Khurram Javed and Richard S Sutton. The Big World Hypothesis and its Ramifications for Artificial Intelligence. 2024.

Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

Amirmohammad Karimi, Jun Jin, Jun Luo, A. Rupam Mahmood, Martin Jagersand, and Samuele Tosatto. Dynamic Decision Frequency with Continuous Options. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7545–7552, October 2023. DOI: 10.1109/IROS55552.2023.10342408. URL https://ieeexplore.ieee.org/document/10342408. ISSN: 2153-0866.

Mark Kelly and Andrew B. Barron. The best of both worlds: Dual systems of reasoning in animals and AI. *Cognition*, 225:105118, August 2022. ISSN 0010-0277. DOI: 10.1016/j.cognition.2022.105118. URL https://www.sciencedirect.com/science/article/pii/S0010027722001068.

Mehdi Keramati, Amir Dezfouli, and Payam Piray. Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLoS computational biology*, 7(5):e1002055, May 2011. ISSN 1553-7358. DOI: 10.1371/journal.pcbi.1002055.

Wouter Kool, Samuel J. Gershman, and Fiery A. Cushman. Planning Complexity Registers as a Cost in Metacontrol. *Journal of Cognitive Neuroscience*, 30(10):1391–1404, October 2018. ISSN 1530-8898. DOI: 10.1162/jocn_a_01263.

Paul M. Krueger, Falk Lieder, and Thomas L. Griffiths. Enhancing metacognitive reinforcement learningusing reward structures and feedback. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 39(0), 2017. URL https://escholarship.org/uc/item/39s4316w.

Linh Kästner, Johannes Cox, Teham Buiyan, and Jens Lambrecht. All-in-One: A DRL-based Control Switch Combining State-of-the-art Navigation Planners, September 2021. URL http://arxiv.org/abs/2109.11636. arXiv:2109.11636 [cs].

Aravind Lakshminarayanan, Sahil Sharma, and Balaraman Ravindran. Dynamic Action Repetition for Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), February 2017a. ISSN 2374-3468. DOI: 10.1609/aaai.v31i1.10918. URL https://ojs.aaai.org/index.php/AAAI/article/view/10918.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, November 2017b. URL http://arxiv.org/abs/1612.01474. arXiv:1612.01474 [stat].

Sang Wan Lee, Shinsuke Shimojo, and John P. O'Doherty. Neural Computations Underlying Arbitration between Model-Based and Model-free Learning. *Neuron*, 81(3):687–699, February 2014. ISSN 0896-6273. DOI: 10.1016/j.neuron.2013.11.028. URL https://www.cell.com/neuron/abstract/S0896-6273(13)01125-2.

Falk Lieder and Thomas L. Griffiths. Strategy selection as rational metareasoning. *Psychological Review*, 124(6):762–794, 2017. ISSN 1939-1471. DOI: 10.1037/rev0000075.

Falk Lieder, Amitai Shenhav, Sebastian Musslick, and Thomas Griffiths. *Rational metareasoning and the plasticity of cognitive control*. February 2018. DOI: 10.13140/RG.2.2.24500.14721.

Rohin Manvi, Joey Hong, Tim Seyde, Maxime Labonne, Mathias Lechner, and Sergey Levine. Zero-Overhead Introspection for Adaptive Test-Time Compute, December 2025. URL http://arxiv.org/abs/2512.01457. arXiv:2512.01457 [cs].

Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. Control Frequency Adaptation via Action Persistence in Batch Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 6862–6873. PMLR, November 2020. URL https://proceedings.mlr.press/v119/metelli20a.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs with Preference Data, February 2025. URL http://arxiv.org/abs/2406.18665. arXiv:2406.18665 [cs].

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph

Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. OpenAI o1 System Card, December 2024. URL http://arxiv.org/abs/2412.16720. arXiv:2412.16720 [cs].

Adrian Orenstein, Jessica Chen, Gwyneth Anne Delos Santos, Bayley Sapara, and Michael Bowling. Toward Agents That Reason About Their Computation, October 2025. URL http://arxiv.org/abs/2510.22833. arXiv:2510.22833 [cs].

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning When to Plan: Efficiently Allocating Test-Time Compute for LLM Agents, September 2025. URL http://arxiv.org/abs/2509.03581. arXiv:2509.03581 [cs].

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Amir Hossain Raj, Zichao Hu, Haresh Karnan, Rohan Chandra, Amirreza Payandeh, Luisa Mao, Peter Stone, Joydeep Biswas, and Xuesu Xiao. Rethinking Social Robot Navigation: Leveraging the Best of Two Worlds. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16330–16337, May 2024. DOI: 10.1109/ICRA57147.2024.10611710. URL https://ieeexplore.ieee.org/abstract/document/10611710.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1):361–395, May 1991. ISSN 0004-3702. DOI: 10.1016/0004-3702(91)90015-C. URL https://www.sciencedirect.com/science/article/pii/000437029190015C.

C. Nicolò De Sabbata, Theodore R. Sumers, Badr AlKhamissi, Antoine Bosselut, and Thomas L. Griffiths. Rational Metareasoning for Large Language Models, June 2025. URL http://arxiv.org/abs/2410.05563. arXiv:2410.05563 [cs].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347 [cs].

Sahil Sharma, Aravind Srinivas, and Balaraman Ravindran. Learning to Repeat: Fine Grained Action Repetition for Deep Reinforcement Learning, September 2020. URL http://arxiv.org/abs/1702.06054. arXiv:1702.06054 [cs].

Vishnu Dutt Sharma, Jeongran Lee, Matthew Andrews, and Ilija Hadžic. Hybrid Classical/RL Local Planner for Ground Robot Navigation. 2025.

Farzaneh Sheikhnezhad Fard and Thomas P. Trappenberg. A Novel Model for Arbitration Between Planning and Habitual Control Systems. *Frontiers in Neurorobotics*, 13, July 2019. ISSN 1662-5218. DOI: 10.3389/fnbot.2019.00052. URL https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2019.00052/full.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, August 1999. ISSN 0004-3702. DOI: 10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Kevin A. Wang, Jerry Xia, Stephen Chung, and Amy Greenwald. Dynamic Thinker: Optimizing Decision-Time Planning with Costly Compute. April 2025. URL https://openreview.net/forum?id=yGglBJ1pjZ.

Taokai Xia and Hui Chen. A Survey of Autonomous Vehicle Behaviors: Trajectory Planning Algorithms, Sensed Collision Risks, and User Expectations. *Sensors (Basel, Switzerland)*, 24(15): 4808, July 2024. ISSN 1424-8220. DOI: 10.3390/s24154808. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC11314818/.

Kenny Young, Aditya Ramesh, Louis Kirsch, and Jürgen Schmidhuber. The Benefits of Model-Based Generalization in Reinforcement Learning, July 2023. URL http://arxiv.org/abs/2211.02222. arXiv:2211.02222 [cs].

Supplementary Materials

The following content was not necessarily subject to peer review.

A Appendix 1: Hyperparameters

	Box Push	DoorKey	Fetch	Maze	Navigation
Reactive Pretraining					
Steps collected	10,000	10,000	2,000	10,000	10,000
Epochs	30	30	30	30	30
Loss	CE	CE	MSE	CE	MSE
Planner					
Task horizon	30	25	30	12	30
Short / Med / Long plan	5/10/15	4/8/12	5/10/15	2/4/6	5/10/15

Table 4: Environment-specific hyperparameters for reactive pretraining and planner configuration. CE = Cross-Entropy, MSE = Mean Squared Error. Batch size 64 across all environments.

Hyperparameter	Value
Joint-Training Reactive Policy	
Train frequency	2048 steps
Buffer size	10,000
Epochs / Batch size	3 / 64
Learning rate	1e-4

Table 5: Joint-training hyperparameters.

B Reactive Policy Architecture

Here we detail the architectures used for the reactive policy.

Actor Type	Obs. Type	Action	Architecture
Image	Image	Discrete	*CNN:* Conv(C$\to$16, 3$\times$3) $\to$ ReLU $\to$ MP(2) *MLP:* FC(n $\to$512) $\to$ ReLU $\to$ FC(512$\to$ d)
Vector	Tensor	Discrete	*MLP:* FC(d_{in} $\to$512) $\to$ ReLU $\to$ FC(512$\to$512) $\to$ ReLU $\to$ FC(512$\to$ d)
Image-Cont	Image	Continuous	*CNN:* [Conv $\to$ ReLU $\to$ MP(2)]$\times$3 $\to$ AAP(8$\times$8) *Shared:* FC(n $\to$1024) $\to$ ReLU $\to$ FC(1024$\to$512) *Head:* FC(512$\to$ a) $\to$ Tanh

Table 6: Model-free actor network architectures. Image is used for Maze. Vector is used for BoxPush and DoorKey. Image-Cont is used for Fetch and Navigation. MP = MaxPool2d, AAP = AdaptiveAvgPool2d, FC = fully-connected, d = `features_dim`, a = `action_dim`, n = CNN output size.

C Separated Environment Ablations

Here we show the results in Figure 3 non-averaged over environments.

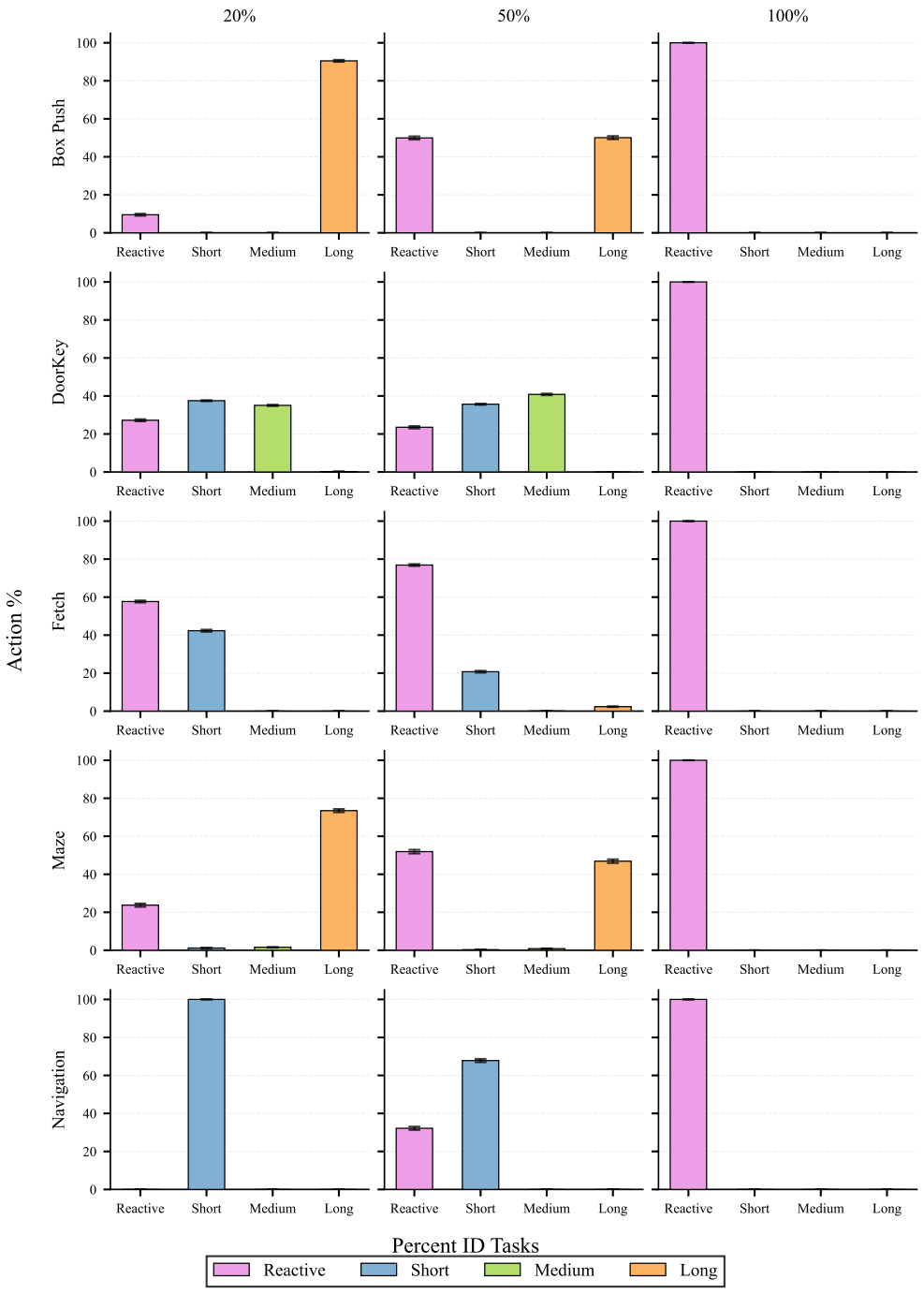


Figure 5: **Reactive policy in-distribution percentage experiment.** Individual environment results when varying the percent of in-distribution tasks for the reactive policy.

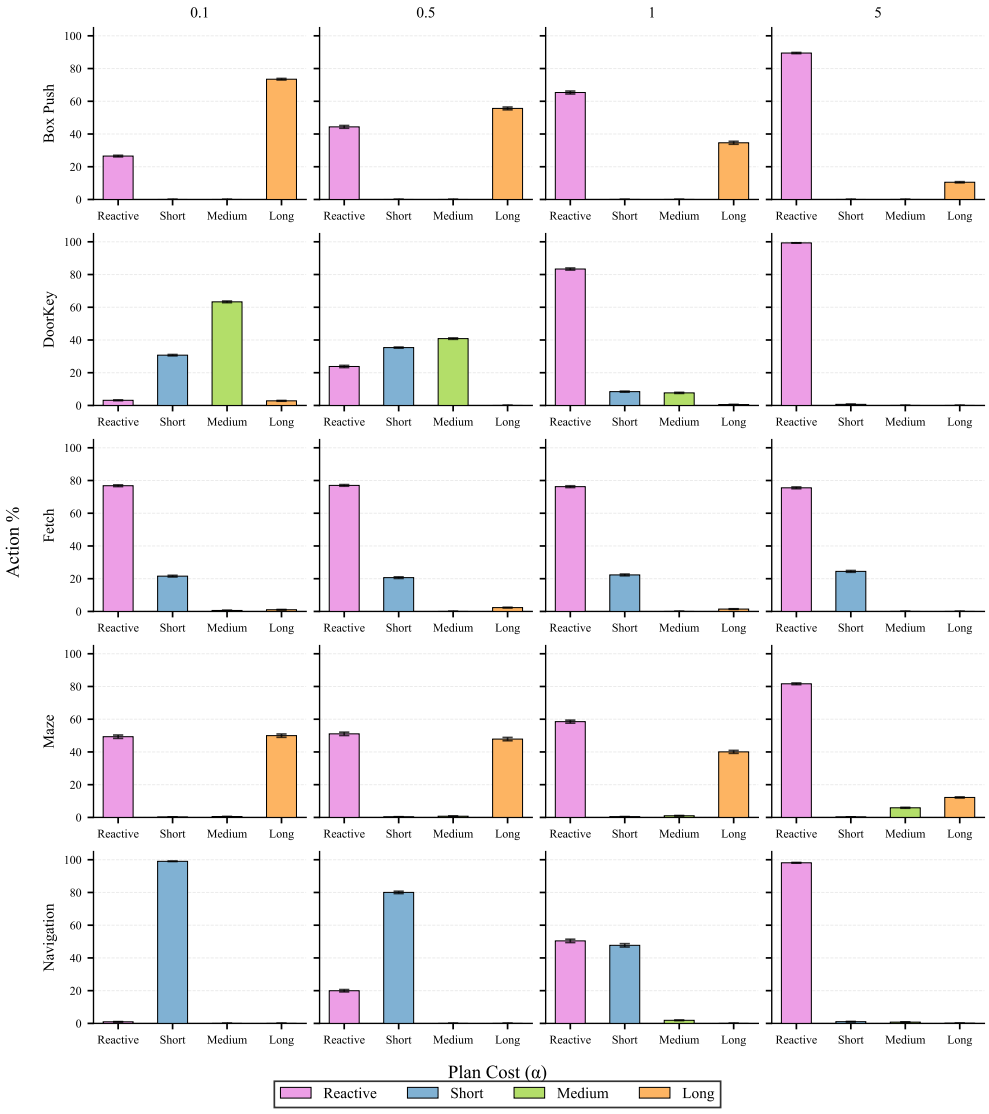


Figure 6: **Plan cost experiment.** Individual environment results when varying the planning cost α.

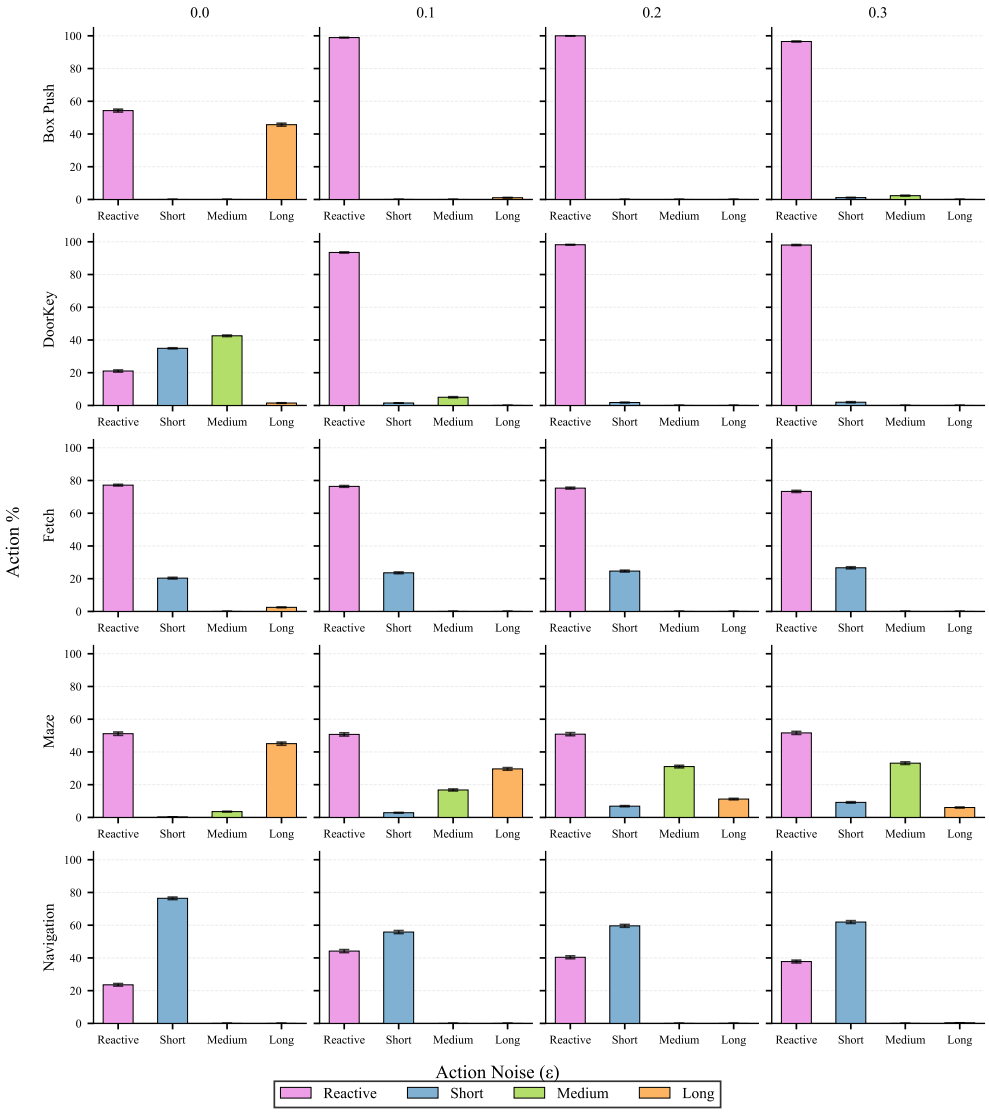


Figure 7: **Action stochasticity experiments.** Individual environment results when varying the environment stochasticity.