EXPLORATORY PROMPTING FOR IN-CONTEXT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context reinforcement learning pretrains a sequence model on many tasks' interaction data and deploys it frozen, expecting it to improve on a new task from the episodes it adds to its context. We show that a cold start can trap the agent in that context. Acting with its mean action, the model plays a default episode from an empty context. The context it then writes can repeat or amplify that episode, although the episode's rewards still carry the task. We evaluate seven in-context RL methods on ten task families and dissect the trap in the family where a stuck trial is unambiguous. Context swaps in frozen models show that the model reads the task from the rewards of an episode it did not write. A random episode releases almost every stuck trial, and relabeling its rewards steers the agent toward the relabeled task. Our remedy, exploratory prompting, needs neither retraining nor a demonstration: the agent spends one episode acting at random. When its context is a single prompt episode, it keeps that episode as the prompt only if the return it induces is at least the return its own first episode induces. For four methods, exploratory prompting removes most stuck trials and raises the mean final return in 34 of 40 (family, method) pairs by mean gain, at some cost in cumulative return. Separately, for single-prompt methods, dropping the prompt at random during a short fine-tuning stage also removes most stuck trials, whereas step-matched continued training does not. We argue that part of the reported difference between in-context RL methods reflects how they start, not how well they learn.

1 INTRODUCTION

In-context reinforcement learning (ICRL) pretrains a sequence model on interaction data from many tasks and deploys it frozen (Laskin et al., 2023; Lee et al., 2023; Moeini et al., 2025). On a new task it acts, appends its own episodes to its context, and should improve without a weight update. Some methods prompt it with a demonstration (Xu et al., 2022); others start from an empty context (Laskin et al., 2023; Lee et al., 2023; Wu et al., 2025; Liu et al., 2026). Polubarov et al. (2026) call the second form cold-start evaluation. They report that at scale it matches the first on eight of their ten domains but trails it on the two manipulation benchmarks, without isolating a mechanism. We ask how a cold start can fail in continuous control, and how to fix it without retraining.

A cold-start trial is often decided by its first episodes. Many continuous-control methods act with their mean action (Xu et al., 2022; Wang et al., 2024; Wu et al., 2025; Polubarov et al., 2025), so an empty context yields one deterministic default policy, the same for every task. We study models that read their own whole episodes: AD-lineage models read a buffer of four of their episodes sorted by return, and DPT-lineage models read their previous episode as a prompt (Section 3). When the default episode is poor, the context the model writes can hold it there. We call such a behavior an *attractor*: stuck AD-lineage agents keep standing still, and stuck DPT-lineage agents run ever faster past their target (Figures 1 and 2). In the family where a stuck trial is unambiguous, each of the four AD- and DPT-lineage methods leaves a substantial share of its held-out trials stuck (Section 4), although the rewards of the self-written context carry the task. Context swaps on frozen checkpoints show the model reading the task from the rewards of a random episode: that episode releases almost every stuck trial, and relabeled rewards steer the agent toward the relabeled task (Figure 3).

The diagnosis suggests a remedy that needs neither retraining nor a demonstration: *exploratory prompting* (EP, Section 5) gives the frozen model one episode it did not write and lets the return

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

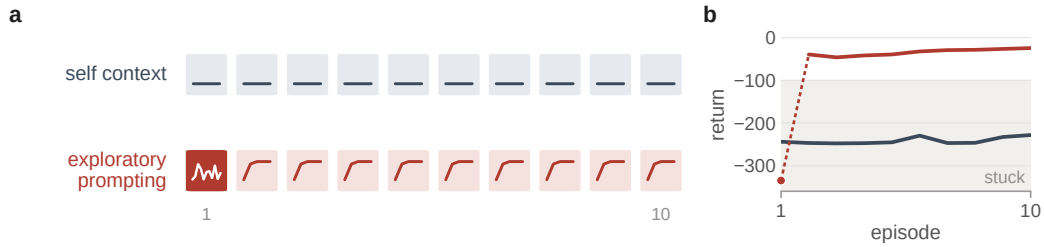


Figure 1: **The context trap and exploratory prompting.** (a) Schematic of the ten episodes of one trial; each tile sketches the forward speed within an episode. Under the self context the model writes a first episode that stands still, and every later episode repeats it. Under exploratory prompting (drawn for the AD lineage) the first episode is random, the model reads the task from its rewards, and the later episodes run. (b) Return per episode of one HalfCheetah-Vel trial of T2MIR-AD under each context, with the same checkpoint, task and reset; the band marks stuck returns, below -100.

that episode induces decide whether to keep it. The agent spends one episode acting uniformly at random. The DPT lineage keeps it as the prompt only if the return it induces is at least the return that the agent's own first episode induces (Algorithm 1); the AD lineage admits it into the return-sorted buffer. EP's mean paired gain in final return is positive in 34 of 40 (family, method) pairs, and a hierarchical bootstrap puts its 95% interval above zero in 17 pairs and below zero in 1 (Section 7). In the DPT lineage, EP costs cumulative return. When retraining is possible, prompt dropout in a short fine-tuning stage removes most stuck trials, whereas step-matched continued training does not (Section 7.3).

In-context learners can be misled by their own history (Zhang et al., 2025; Krishnamurthy et al., 2024; Monea et al., 2025). Unlike the sampling-induced ambiguity of Zhang et al. (2025), this trap arises under deterministic decoding, in episode 1. Our contributions are: (i) a cold-start evaluation of seven ICRL methods on ten families under one harness that keeps each method's own context rule, with a dissection of two lineage-specific attractors on HalfCheetah-Vel and context swaps showing that a frozen model reads the task from the rewards of an episode it did not write (Section 4); (ii) **exploratory prompting**, a training-free, demonstration-free rule that selects its prompt by the return it induces, with an explore-then-commit bound for that selection (Sections 5 and 6), and **prompt-dropout fine-tuning**, tested against step-matched continued training (Section 7.3); and (iii) evidence that under exploratory prompting the differences between methods shrink on most families (Section 7.1).

2 Related work

In-context reinforcement learners. Algorithm Distillation (AD) trains a sequence model (Chen et al., 2021; Janner et al., 2021) on across-episode learning histories (Laskin et al., 2023). Later work builds such histories by noise injection (Zisman et al., 2024), filters them for steady improvement (Chen et al., 2025), adds planning (Son et al., 2025), changes the backbone (Beaussant & Mounsif, 2025) or scales across domains (Polubarov et al., 2025). The Decision-Pretrained Transformer (DPT) predicts the optimal action from an in-context dataset (Lee et al., 2023). Follow-ups distill its labels from random-policy rollouts (Chen & Paternain, 2025) or advantage-weighted suboptimal data (Dong et al., 2025), and scale it across domains with a flow-matching head (Polubarov et al., 2026). Related agents condition on a trajectory prompt (Xu et al., 2022; Wang et al., 2024) or learn in-context values (Liu et al., 2026); others add mixture-of-experts routing (Wu et al., 2025). Lin et al. (2024) bound the ICRL error by an expert-to-offline distribution ratio; Moeini et al. (2025) survey the area. Appendix I gives details.

Failures of self-generated context. In gridworlds, stochastic sampling makes the model misread its context's learning stage (Zhang et al., 2025). Language models learning from reward in context show suffix failures and instability (Krishnamurthy et al., 2024; Monea et al., 2025). Polubarov et al. (2026) report the cold-start gap at scale. Our attractor resembles context shift in offline meta-RL (Gao et al., 2023; Pong et al., 2022) and latching, copycat behavior and compounding error in imitation (Swamy et al., 2022; Wen et al., 2020; Ross et al., 2011). Distinct from the MDP ambiguity of offline meta-RL (Dorfman et al., 2021), it arises although the context carries the task.

Exploration, prompt selection and conditioning dropout. Meta-RL agents learn to explore or infer tasks (Duan et al., 2016; Stadie et al., 2018; Liu et al., 2021; Zintgraf et al., 2020; 2021; Zhang et al., 2021; Rakelly et al., 2019; Dai et al., 2024; Beck et al., 2025); EP trains nothing. Prior work tunes, generates, selects or retrieves decision-transformer context (Hu et al., 2026; 2024; Wang et al., 2024; Schmied et al., 2026). Closest to our rule, Rietz et al. (2025) select prompts with a bandit on induced return; we compare the agent's own first episode against an exploratory one (Lattimore & Szepesvári, 2020). Prompt dropout is conditioning dropout (Ho & Salimans, 2022; Ajay et al., 2023; Frans et al., 2025; Octo Model Team et al., 2024) on the context.

3 In-context reinforcement learning from a cold start

A task family is a set of MDPs M_θ with shared state and action spaces, indexed by a task parameter $\theta \in \Theta$ from a prior $\mathcal{P}$. An episode is $\tau = (s_t, a_t, r_t)_{t=1}^{H}$ with horizon $H = 200$ and return $G(\tau) = \sum_t r_t$. An ICRL model is pretrained offline on contexts from $\mathcal{D}_{\text{pre}}$, built from multi-task learning histories. It is then frozen and deployed as $\pi_\phi(a \mid s, C)$ for a context C of earlier episodes. As in its release, it acts with the mean of its action head, $a_t = \mu_\phi(s_t, C)$. A trial on a held-out task has $K = 10$ episodes, and a context rule U writes each episode into the context:

$$C_1 = \varnothing, \qquad \tau_k = F_\theta(C_k; \xi_k), \qquad C_{k+1} = U(C_k, \tau_k), \tag{1}$$

where F_θ maps a context to the episode it produces on M_θ, and ξ_k is the reset noise. The empty first context is the cold start. At deployment, the two lineages in most of our grid differ only in U. The AD lineage keeps a buffer of $M = 4$ of its episodes sorted by return, lowest first: $U_{\text{AD}}(C, \tau) = \text{sort}_{G\uparrow}(\text{last}_M(C \oplus \tau))$. Here last_M drops the lowest-return entry of C, not the oldest: the newest episode always enters and the buffer resembles an improving learning history. Our AD build inherits this rule from the T2MIR release (Wu et al., 2025). The original AD keeps a queue of recent transitions (Laskin et al., 2023). The DPT lineage follows the T2MIR-DPT release (Wu et al., 2025) and prompts each episode with its previous one, held fixed throughout: $U_{\text{DPT}}(C, \tau) = \tau$. The original DPT appends episodes up to its context length (Lee et al., 2023).

Families, data and methods. We use six MuJoCo locomotion families (HalfCheetah-Vel, Hopper-Mass, Hopper-Friction, Walker-Mass, Walker-Friction, Humanoid-Dir; Todorov et al., 2012) and four MetaWorld manipulation families (Reach, Push, Pick-Place, Soccer; Yu et al., 2020). The MetaWorld observations exclude goal coordinates. Each family has 20 tasks; we hold out 5, one from each block of four consecutive tasks. The split is in-distribution on purpose: the failure we study needs no extrapolation. Each task's offline data are the evaluation rollouts of one SAC run (Haarnoja et al., 2018) at many checkpoints, in checkpoint order, so they form a learning history. Neither lineage reads that order in training (Appendix B). We evaluate T2MIR-AD and T2MIR-DPT (Wu et al., 2025); AD (Laskin et al., 2023) and DPT (Lee et al., 2023), built as T2MIR without mixture-of-experts layers; Prompt-DT (Xu et al., 2022); Meta-DT (Wang et al., 2024); and S-ICQL (Liu et al., 2026). Each is trained with its released code on our data and scored by one harness that keeps its own context rule (Appendix B). As a reference that does not start cold, Prompt-DT also runs with a 5-step segment of a top-decile offline episode of the held-out task ("oracle demo").

Harness and metrics. The harness re-seeds the environment per (task, trial). So methods see the same initial states. MuJoCo families run 10 trials per held-out task. MetaWorld, with deterministic resets, runs one. Every cell has six training seeds, and $\pm$ denotes a standard deviation unless marked s.e. Our primary metric is the last-5 return $\frac{1}{5}\sum_{k=6}^{10} G(\tau_k)$, the mean over the last five episodes, when adaptation has ended. The AUC $\frac{1}{10}\sum_{k=1}^{10} G(\tau_k)$ also counts the early episodes. On MetaWorld, success@0.10 is the share of episodes 6 to 10 whose minimum object-to-goal distance (end effector on Reach) is at most 0.10 m, and success@flag uses MetaWorld's success flag. The oracle demo is a reference, not a ceiling (Appendix C).

4 The cold-start attractor

We make the failure precise on HalfCheetah-Vel, where it is easiest to see. The reward is $r_t = -|v_t - \theta| - c_{\text{ctrl}}\|a_t\|^2$ for forward speed v_t and target speed θ. So an episode in which the robot stays in place earns

$$G(\tau) \approx -H\theta = -200\,\theta. \tag{2}$$

Table 1: **Cold-start methods leave many HalfCheetah-Vel trials stuck.** Share of held-out trials (50 per seed) with last-5 return below -100 under each method's own context rule; last column, last-5 return. Oracle row: Prompt-DT with the oracle demo. Meta-DT: 40 epochs, about 461k steps (release: 200k steps; Appendix B).

	stuck rate per seed							
Method	s0	s1	s2	s3	s4	s5	mean	last-5 return
T2MIR-AD	0.52	0.26	0.02	0.36	0.32	0.06	0.257	-109.7 ± 63.3
AD	0.54	0.38	0.02	0.22	0.48	0.20	0.307	-128.1 ± 50.3
T2MIR-DPT	0.06	0.08	0.04	0.10	0.24	0.08	0.100	-61.3 ± 14.0
DPT	0.30	0.00	0.18	0.24	0.56	0.12	0.233	-92.4 ± 35.6
Prompt-DT	0.52	0.52	0.52	0.60	0.58	0.44	0.530	-144.6 ± 11.8
Meta-DT	1.00	1.00	1.00	1.00	1.00	1.00	1.000	-253.3 ± 39.6
S-ICQL	0.00	0.00	0.24	0.06	0.00	0.00	0.050	-44.3 ± 14.7
Prompt-DT (oracle demo)	0.00	0.02	0.02	0.00	0.00	0.00	0.007	-50.3 ± 2.0

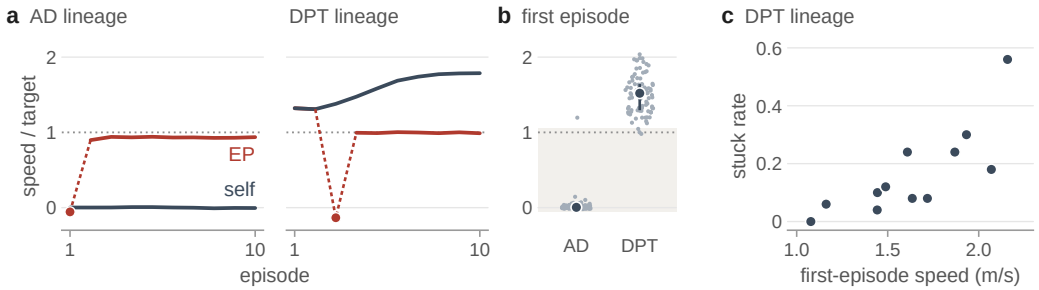


Figure 2: **Two ways of getting stuck on HalfCheetah-Vel.** (a) Speed over target, v/θ, per episode under self and EP, for one stuck trial each of T2MIR-AD seed 0 (AD lineage) and DPT seed 4 (DPT lineage), the one with the median late speed (all: Figure 14); dot, the random episode; dotted, the target. (b) First-episode v/θ of every stuck trial whose rerun reproduces the grid's episode-1 return to within 1 (168 of 169 AD-lineage, 88 of 100 DPT-lineage); dot and bar, median and quartiles; band, the range of the 7,410 SAC episodes (Figure 17a); standing AD-lineage starts lie inside it but below every SAC episode in mean per-step speed $|v_t|$. (c) Twelve DPT-lineage checkpoints: stuck rate against median first-episode speed (Spearman $\rho = 0.82$).

On the held-out targets $\theta \in [1.2, 3.0]$ m/s this is -240 to -600, against -50.3 for the oracle demo.

Definition 1 (Stuck, standing and overshooting)**.** A HalfCheetah-Vel trial is *stuck* if its last-5 return is below -100, a threshold specific to this family. An episode *stands* if its mean forward speed is below 0.1 m/s in absolute value, and *overshoots* if its mean speed exceeds the target speed θ.

Returns alone cannot tell standing from overshooting: running at 2θ also returns -200θ. We therefore classify episodes from logged speeds.

Prevalence. Five of the seven methods leave between one in ten and about half of their held-out trials stuck; Meta-DT leaves all of them stuck; and S-ICQL, which re-encodes its previous episode, leaves one in twenty (Table 1). Separately, no stuck trial of Prompt-DT earns the standing return in every episode; its returns fit one fixed speed whatever the target (Appendix C). For Prompt-DT, with the same weights, context quality alone moves the stuck rate from 0.530 with its own prompt to 0.007 with the oracle demo. The rate varies more across seeds of one method (0.00 to 0.56 for DPT) than between the four AD- and DPT-lineage means (0.100 to 0.307). Few-seed comparisons on this family therefore partly measure how often first episodes fall into an attractor.

Two attractors. Trajectory logs of 24 checkpoints show two failures (Figure 2). We call a behavior an *attractor* when the self-written context keeps the agent in it. Of the stuck trials whose rerun matches the grid, the AD-lineage ones stand in episode 1 (165 of 168). A buffer of such episodes tends to keep the agent standing. The DPT-lineage ones overshoot, and none of the 88 stands. Their empty-context episode runs faster than the target, and after episode 2 each self-written prompt makes the next episode faster, so the return declines. Most DPT-lineage checkpoints run their first episode at nearly the same speed whatever the target. This default speed ranks the checkpoints' stuck rates (Spearman $\rho = 0.82$, 12 checkpoints). So in this lineage much of the seed spread in Table 1 tracks one number. Both kinds of first episode are atypical of the pretraining data: 85 of the 88 DPT-lineage

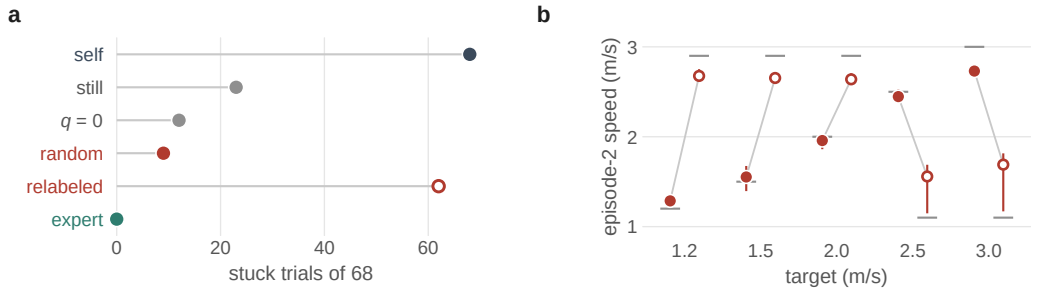


Figure 3: **The frozen model reads the rewards of its context episode** (HalfCheetah-Vel; seven stuck-prone checkpoints: T2MIR-AD seeds 0, 3, 4, AD seed 1, DPT seeds 0, 4, T2MIR-DPT seed 4; 25 shared resets per arm). (a) Of the 68 trials stuck under the self context, those that remain stuck when the first context episode is replaced. *still*: zero actions; $q = 0$: the first-checkpoint episode of the task's own SAC run (both stand on average); *random*: a uniformly random episode; *relabeled*: the same random episode with rewards for a wrong target θ' (its trials count as stuck because they run at speed θ'); *expert*: a SAC-expert episode. (b) Median episode-2 speed after the random episode with true (filled) or relabeled (hollow) rewards; bars, interquartile range over the 35 trials per target; gray ticks, the target the rewards describe.

ones run faster relative to the target than all 7,410 SAC episodes, and 164 of the 168 AD-lineage ones move less per step than every SAC episode (Appendix F.4). Atypicality does not decide the outcome: many first episodes that escape are as atypical, and across AD-lineage checkpoints the share of standing starts that stay stuck ranges from 3% to 93% (Appendix F).

Swapping one context episode. We keep the weights and resets of seven stuck-prone checkpoints, four AD-lineage and three DPT-lineage, and replace only the first context episode (Figure 3). Any change in outcome therefore comes from that episode. The 68 self-stuck trials fall in 24 checkpoint-task cells. A uniformly random episode releases 59 of the 68 trials stuck under the self context, nearly as many as a SAC-expert episode (68 of 68). Yet it returns about -450: the return a context induces does not follow its own return. The model reads the random episode's rewards. Relabeled for a wrong target θ', they move the next episode's speed toward θ' in 175 of 175 paired trials, and in 170 that speed is closer to θ' than to θ. A standing context traps only the AD lineage, and not on every checkpoint. On T2MIR-AD seeds 0, 3 and 4 and AD seed 1, a zero-action episode leaves 21, 12, 2 and 9 of the 25 replayed trials stuck, against 12, 8, 9 and 10 under the self context. On the three DPT-lineage checkpoints, whose stuck trials overshoot, it leaves none.

What the self-written context lacks. Not information. Every reward of a standing episode equals $-\theta$ up to the control cost, so the self-written context identifies the task (Appendix A). The same frozen model reads the task from almost any moving episode it did not write. An episode from anywhere past the first tenth of the task's SAC run leaves 0 of 25 trials stuck on every checkpoint, whatever its own return (Appendix F.3). Where its pretraining data have negligible mass, little constrains the model (Appendix A). There, the self chain, the sequence of self-written contexts, repeats (AD lineage) or amplifies (DPT lineage) its first episode. A random episode is far from the data too (Appendix F.4), yet the model reads it.

5 Exploratory Prompting

Exploratory prompting changes only the context rule U and the actions of one episode; the weights, the deterministic head and the rest of each released rule are unchanged. We apply it to the four methods of the DPT and AD lineages, whose deployment context consists of their own whole episodes. We do not apply EP to Prompt-DT, which is pretrained on top-decile prompts and loses return under a random one (Xu et al., 2022), or to Meta-DT and S-ICQL, which need other EP rules.

DPT lineage. Algorithm 1 keeps the first two episodes of the released rule and explores at episode 3. Episode 4 tests the random episode as a prompt: the rule commits to it only if the return it induces is at least the return that the agent's own first episode induced in episode 2. *A whole exploratory episode, not per-step noise:* on HalfCheetah-Vel, it replaces the self-written prompt with one whose rewards the model reads (Figure 3). A Gaussian or ε-greedy episode under the same rule also

Algorithm 1 Exploratory prompting, DPT lineage (`always3`). $F_\theta(C;\xi)$ rolls out one episode with the frozen head $a_t = \mu_\phi(s_t, C)$ and reset noise ξ; $F_\theta^{\text{rand}}(\xi)$ one with i.i.d. uniform actions on $\mathcal{A} = [-1,1]^{d_a}$.

Require: frozen model μ_ϕ, unknown task θ, trial length K

$C_1 \leftarrow \varnothing$; $\tau_1 \leftarrow F_\theta(C_1;\xi_1)$ ▷ cold start, released rule
$C_2 \leftarrow \tau_1$; $\tau_2 \leftarrow F_\theta(C_2;\xi_2)$; $v_{\text{self}} \leftarrow G(\tau_2)$ ▷ induced by τ_1
$\tau_3 \leftarrow F_\theta^{\text{rand}}(\xi_3)$ ▷ exploratory episode
$C_4 \leftarrow \tau_3$; $\tau_4 \leftarrow F_\theta(C_4;\xi_4)$; $v_{\text{exp}} \leftarrow G(\tau_4)$ ▷ induced by τ_3
if $v_{\text{exp}} \geq v_{\text{self}}$ **then** ▷ commit to the exploratory prompt
  for $k = 5,\dots,K$: $C_k \leftarrow \tau_3$; $\tau_k \leftarrow F_\theta(C_k;\xi_k)$
else ▷ resume the self chain after episode 2
  $C_5 \leftarrow \tau_2$; **for** $k = 5,\dots,K$: $\tau_k \leftarrow F_\theta(C_k;\xi_k)$; $C_{k+1} \leftarrow \tau_k$
end if

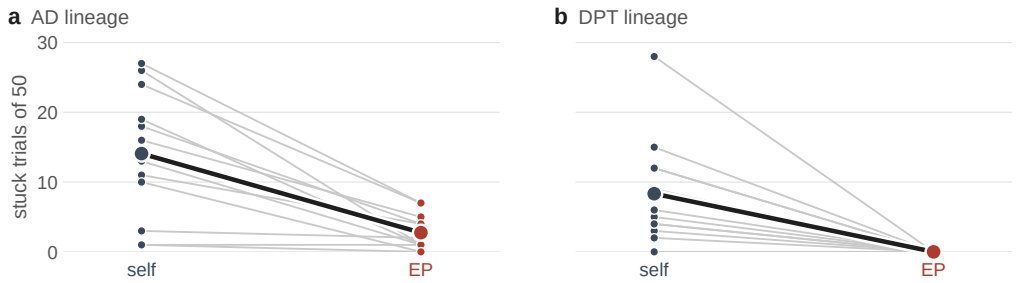


Figure 4: **One exploratory episode releases trapped trials.** Stuck HalfCheetah-Vel trials (last-5 return below -100) under the agent's own context (self) and under EP, which is a random first episode in the AD lineage and `always3` in the DPT lineage. Thin lines, the 12 checkpoints of each lineage (50 held-out trials each); thick line, their mean. EP releases 152 of the 169 self-stuck AD-lineage trials and makes 16 others stuck; in the DPT lineage it releases all 100.

releases most stuck trials, though fewer than a uniform episode does (Appendix E). *Selection by induced return:* for a frozen model a prompt is worth the return it induces, not the return of the episode it contains. A random episode scores poorly by construction but can induce a high return; keeping it unconditionally (`rand1keep`) loses on six of twenty DPT-lineage pairs. *Two own episodes first:* a random first episode displaces a self chain that would improve on its own. Exploring only when episode 2 fails to improve misses chains that improve briefly and then fall (Appendix E).

AD lineage. The AD context is a return-sorted buffer of four episodes that the model reads jointly, so there is no single prompt to select. EP makes episode 1 uniformly random and changes nothing else; the buffer places it by its return. Exploring at episode 3 instead (`rand3`) gains less (Figure 6b). On HalfCheetah-Vel, one exploratory episode leaves almost no stuck trial in either lineage (Figure 4).

Prompt-dropout fine-tuning. As a second, independent remedy, prompt dropout is conditioning dropout (Ho & Salimans, 2022; Octo Model Team et al., 2024) applied to the in-context prompt. When retraining is possible, we fine-tune each DPT-lineage checkpoint with its own loss for 20k steps. The prompt is masked out of attention with probability 0.5 for the first 10k steps and 0.2 afterward, so the model also learns to act without it. The control runs the same 20k steps without masking. Deployment is unchanged (Appendix G).

6 Fixed points, coverage and prompt selection

The statements below make precise what Section 4 calls an attractor, what pretraining guarantees about a self-written context, and what one test episode per prompt guarantees; Appendix A proves them. Cold-start deployment iterates (1) with a deterministic head and dynamics.

Lemma 1 (Fixed points of cold-start deployment)**.** *Let $\mathfrak{S}$ be a set of contexts with $U(C, F_\theta(C;\xi)) \in \mathfrak{S}$ for every $C \in \mathfrak{S}$ and reset draw ξ. If $C_k \in \mathfrak{S}$, then every later context lies in $\mathfrak{S}$ and every later episode is produced by a context in $\mathfrak{S}$, whatever the resets. With a deterministic reset and $\mathfrak{S} = \{C^\star\}$, that is $U(C^\star, F_\theta(C^\star)) = C^\star$, the episode repeats, $\tau_j = \tau_k$ for every $j \geq k$: the return*

Table 2: **Exploratory prompting against the self context.** Last-5 return on held-out tasks, mean ± std over six seeds. *self*: released rule, empty start; *+EP*: `always3` (Algorithm 1) in the DPT lineage, a random first episode in the AD lineage. Bold: paired win (mean gain above one s.e.). AUC and s.e.: Table 8. Mean paired AUC change −32.7 (DPT), +48.1 (AD).

	T2MIR-AD		AD		T2MIR-DPT		DPT	
Family	self	+EP	self	+EP	self	+EP	self	+EP
HalfCheetah-Vel	-110 ± 63	$\mathbf{-52 \pm 5}$	-128 ± 50	$\mathbf{-51 \pm 7}$	-61 ± 14	$\mathbf{-39 \pm 3}$	-92 ± 36	$\mathbf{-38 \pm 2}$
Hopper-Mass	477 ± 176	$\mathbf{572 \pm 15}$	505 ± 72	537 ± 29	595 ± 7	$\mathbf{607 \pm 3}$	601 ± 15	$\mathbf{605 \pm 9}$
Hopper-Friction	559 ± 49	571 ± 29	583 ± 12	581 ± 7	579 ± 21	$\mathbf{592 \pm 12}$	581 ± 10	$\mathbf{586 \pm 5}$
Walker-Mass	304 ± 181	$\mathbf{529 \pm 106}$	155 ± 220	$\mathbf{525 \pm 59}$	711 ± 40	694 ± 28	721 ± 59	692 ± 43
Walker-Friction	511 ± 173	579 ± 107	382 ± 275	$\mathbf{623 \pm 113}$	769 ± 11	$\mathbf{776 \pm 10}$	772 ± 6	774 ± 7
Humanoid-Dir	694 ± 124	741 ± 35	642 ± 53	644 ± 50	689 ± 70	$\mathbf{727 \pm 36}$	600 ± 67	$\mathbf{636 \pm 69}$
Reach	1028 ± 46	1034 ± 71	976 ± 67	$\mathbf{1045 \pm 118}$	1259 ± 37	$\mathbf{1315 \pm 16}$	1239 ± 53	$\mathbf{1385 \pm 93}$
Push	866 ± 362	$\mathbf{1081 \pm 185}$	959 ± 542	903 ± 342	527 ± 278	$\mathbf{1097 \pm 210}$	988 ± 264	$\mathbf{1165 \pm 197}$
Pick-Place	782 ± 106	754 ± 118	601 ± 289	$\mathbf{754 \pm 94}$	818 ± 126	$\mathbf{938 \pm 94}$	759 ± 107	$\mathbf{900 \pm 80}$
Soccer	631 ± 205	$\mathbf{740 \pm 228}$	701 ± 169	$\mathbf{805 \pm 124}$	1208 ± 252	1264 ± 165	1228 ± 187	1211 ± 198
win / tie / loss	5 / 5 / 0		6 / 4 / 0		8 / 1 / 1		7 / 2 / 1	

is constant and adaptation stops. Under the DPT rule, where $C^\star = \tau^\star$ is one episode, the condition reads $F_\theta(\tau^\star) = \tau^\star$.

Stuck AD-lineage trials that stand in each of episodes 6 to 10 are consistent with such a set; accelerating DPT-lineage chains are not fixed points (Appendix A).

Lemma 2 (Coverage by change of measure)**.** *Let $\mathcal{D}_{\text{pre}}$ and Q be the laws of (task, context) in pretraining and deployment, and $\ell(\theta, C) \geq 0$ a per-context loss, such as the imitation error of μ_ϕ. (i) If $dQ/d\mathcal{D}_{\text{pre}} \leq \kappa$, then $\mathbb{E}_Q[\ell] \leq \kappa\, \mathbb{E}_{\mathcal{D}_{\text{pre}}}[\ell]$. (ii) If $Q(A) > 0 = \mathcal{D}_{\text{pre}}(A)$ for a set A, no finite κ gives such a bound; for open A, in a metric on contexts, a Lipschitz loss shows this.*

Part (ii) is a first-context analogue of an unbounded distribution ratio in Lin et al. (2024). It applies to every deterministic self-written first episode, stuck or not, since its first action has probability zero under a stochastic SAC policy. Pretraining constrains the model there only through nearby data (Appendix A). The lemma explains why an attractor is possible, not which trials fall into it.

Selecting the prompt. Algorithm 1 compares two prompts, the agent's own first episode and the exploratory episode, by one test return each: explore-then-commit with one pull per arm (Lattimore & Szepesvári, 2020, Chapter 6). With σ-sub-Gaussian noise and $\sigma > 0$ it keeps the worse prompt with probability at most $e^{-\Delta^2/(4\sigma^2)}$ for a gap Δ in mean induced return. Deterministic MetaWorld resets give $\sigma = 0$, so the better prompt always wins (Proposition 4(i)). On HalfCheetah-Vel the test returns differ by a median of 8 (T2MIR-DPT) and 14 (DPT) within-trial standard deviations, which only bound σ from below (Appendix A). On the other MuJoCo families, 71% to 84% of trials differ by under two standard deviations; there a wrong pick costs at most the gap in mean induced return per committed episode (Proposition 4(ii)). EP loses only on Walker-Mass, where the self chain still improves.

7 Experiments

EP reuses the self-context evaluation's checkpoints, seeds, held-out tasks and resets. So every comparison is paired by checkpoint, task and trial seed (separate runs; Appendix D). Over the 40 (family, method) pairs, the mean paired gain in last-5 return is positive in 34. A hierarchical bootstrap over seeds, tasks and trials puts its 95% interval above zero in 17 pairs and below zero in 1 (Figure 8). We designed and chose the context rules on the held-out tasks we report. On the training tasks, which played no part in that choice, the mean EP gain of each family and lineage has the same sign as on the held-out tasks in 19 of 20 cases. EP also cuts the HalfCheetah-Vel stuck share there from 22.7% to 6.0% in the AD lineage and from 16.0% to 0.2% in the DPT lineage (Appendix E).

7.1 Main result

EP wins 26 of the 40 pairs, ties 12 and loses 2 (Table 2; DPT 15/3/2, AD 11/9/0). It helps most where the self context stalls (Figure 5). On HalfCheetah-Vel, the DPT-lineage last-5 return rises from −76.9 to −38.7 (12 of 12 checkpoints), above the oracle demo's −50.3. The AD-lineage paired gains there are $+57.4 \pm 27.2$ and $+76.7 \pm 20.4$ (s.e.). The stuck rate falls from 0.100 to

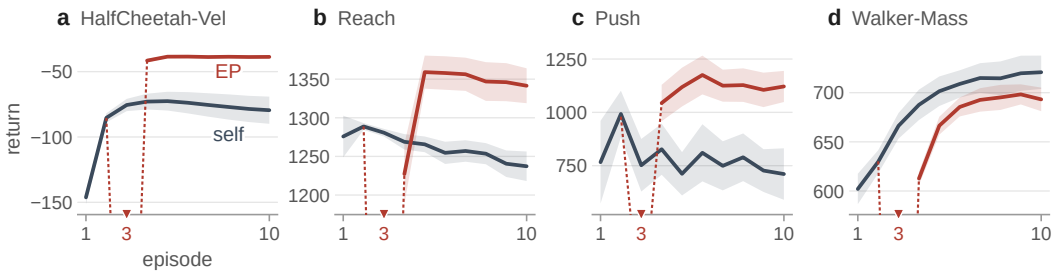


Figure 5: **Per-episode return, DPT lineage, four of the ten families** (all ten: Figure 9; mean over 12 checkpoints, bands ±1 s.e., unpaired). EP (`always3`) makes episode 3 uniformly random (triangle on the bottom edge; mean return −445.0, 355.7, 12.4 and 0.8) and keeps it as the prompt if episode 4 returns at least as much as episode 2. EP ends above self on 12, 12, 10 and 3 of the 12 checkpoints: it lifts later episodes where the self context stalls (a, b, c) and costs return on Walker-Mass (d).

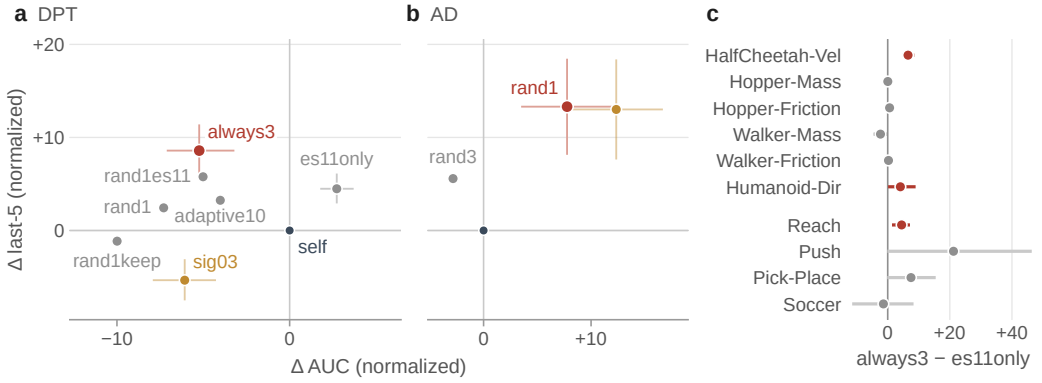


Figure 6: **Training-free context rules against the self context,** paired and normalized per family (100 = the random-to-demo gap). (a, b) Mean change in AUC and last-5 return over the 20 (family, method) pairs of the DPT (a) and AD (b) lineage; origin, self; red, the lineage's EP rule; gold, per-step noise (`sig03`); x scales differ. Bars, 95% bootstrap over seeds with pairs fixed, on the rules the text compares; all rules: Table 10. (c) `always3` minus `es11only` in last-5 return per family, DPT lineage; dot, mean over 12 checkpoints; bar, 95% bootstrap; red, interval above 0.

0.000 (T2MIR-DPT), 0.233 to 0.000 (DPT), 0.257 to 0.043 (T2MIR-AD) and 0.307 to 0.067 (AD). On Reach the DPT-lineage return rises from 1248.6 to 1349.8 (oracle demo 1606.1), success@0.10 from 0.56 to 0.72 and success@flag from 0.01 to 0.17 (Appendix D). On Pick-Place, success@0.10 rises from 0.71 to 0.86; on Push it falls slightly (0.91 to 0.87) while the return rises. Both losses are on Walker-Mass (-16.7 ± 7.0 and -29.0 ± 12.9, s.e.), where the self chain still improves after episode 2 (Section 6). EP also narrows the differences between methods: the range of the four last-5 returns shrinks on eight of ten families (mean 282 to 190). On HalfCheetah-Vel, the lead of each mixture-of-experts method over its plain counterpart (18.4 and 31.1) vanishes (−0.9 and −0.7), so the lead reflected fewer stuck trials (Table 1), not better use of a readable context.

7.2 The cost of the exploratory episode and which part of the rule matters

The exploratory episode and its test cost AUC: over the 20 DPT-lineage pairs, `always3` raises last-5 return by 69.9 but lowers AUC by 32.7 on average (+8.6 and −5.2 normalized; medians +29.1 and −49.6; Figure 6a), with AUC lower in 18 of the 20 pairs. How the random episode is used matters: `rand1`, whose rolling rule drops it after one episode, mostly ties (4/15/1); `rand1keep` loses six pairs; and the rules that compare induced returns gain most. **Selection alone is cheaper without an attractor.** `es11only` picks among the agent's own episodes by induced return. It wins as many pairs as `always3` (15 of 20), loses fewer and raises both last-5 return and AUC (+4.5 and +2.7 normalized), but it releases only 26 of the 100 stuck DPT-lineage HalfCheetah-Vel trials. `always3` beats it on 12 of 12 HalfCheetah-Vel and 11 of 12 Reach checkpoints (Figure 6c). Per-step noise (`sig03`) loses 11 of 20 DPT-lineage pairs. In the AD lineage it matches a random

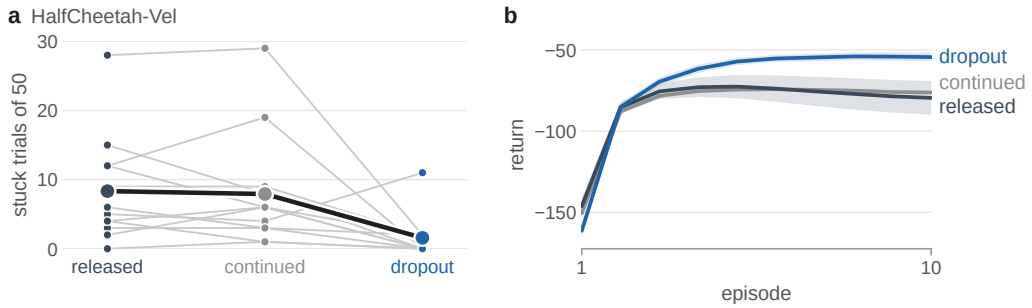


Figure 7: **Prompt dropout against step-matched continued training** (HalfCheetah-Vel, DPT lineage, 12 checkpoints of 50 trials, 20k further steps with or without masking). (a) Stuck trials per checkpoint for the released model, after continued training and after dropout; thin lines, checkpoints; thick line, the mean (pooled: 100, 95 and 19 of 600). (b) Return per episode, mean over checkpoints; bands, ± 1 s.e. for the released model and dropout. All ten families: Figure 18.

first episode on last-5 return ($+13.0$ against $+13.3$ normalized) and beats it on AUC ($+12.3$ against $+7.8$; Figure 6b). It costs no episode but still makes a third of the non-stuck HalfCheetah-Vel trials stuck (Appendix E).

7.3 Prompt-dropout fine-tuning

On HalfCheetah-Vel, prompt dropout cuts stuck trials from 100 to 19 of 600, against 95 for step-matched continued training. Dropout leaves fewer stuck trials than continued training on 11 of 12 checkpoints (sign test $p = 0.006$) and more on one (T2MIR-DPT seed 3, 11 against 4; Figure 7a). Dropout lowers the episode-1 return (Appendix G), but the model then keeps improving from its own episodes after the released model stalls (Figure 7b). In last-5 return against the self context, dropout wins on two families, ties on six and loses on Hopper-Mass and Soccer; on the same checkpoints, pooled per family, EP wins on eight. Dropout costs no episode and has a higher AUC than EP on all ten families at one standard error. Combining the two beats EP alone only on Reach and Push (Figure 18).

8 Discussion and limitations

A single score after ten episodes confounds how well a model learns from a readable context with whether its first episode writes one. ICRL evaluations should state the decoding rule, treat a demonstration prompt as a reference rather than a ceiling, and report the stuck rate, last-5 return, AUC and per-episode curves from cold and exploratory starts. **Limitations.** All our agents act deterministically, with the mean of their action head; sampling heads (Zheng et al., 2022; Lee et al., 2023) may behave differently. The split is in-distribution, all our tasks are continuous control, and our models are small and trained per family. We dissect the mechanism on HalfCheetah-Vel only, though the AD lineage also has a low mode on Walker-Mass, Walker-Friction and Reach, and the DPT lineage on Push (Appendix C). We also do not know which property of a self-written episode keeps the model from reading it. EP covers four of the seven methods, all built on one released code base. It costs AUC in the DPT lineage, though not on average in the AD lineage, and we chose it on the tasks we report.

Conclusion. A frozen in-context learner that starts cold with a deterministic head can be trapped by its own first episode: the context it writes repeats or amplifies that episode, although the rewards in that context carry the task. The same model reads the task from the rewards of a random episode. Exploratory prompting turns this into a remedy that needs no retraining and no demonstration: one random episode, which the DPT lineage keeps as its prompt only if it induces at least the return that the agent's own first episode induces. In the DPT lineage, prompt dropout during a short fine-tuning stage also removes most stuck trials, without spending an episode. We argue that part of the reported difference between in-context RL methods reflects how they start, not how well they learn. Explaining why a model reads a random episode yet not its own first episode may show how to pretrain in-context learners that escape this trap.

AI USE STATEMENT

We used generative AI tools to search the literature, edit language, write and review analysis and evaluation scripts, and audit mathematical statements and proofs. The tools also helped identify places where empirical claims needed narrower wording. We checked every cited source, proof step, numerical statement, code file, and AI-assisted edit against the original papers, the evaluation records, and the released code. We take responsibility for the final content, including all text, claims, proofs, and artifacts that AI assistance helped produce.

ETHICS STATEMENT

This work trains and evaluates sequence models on offline data that reinforcement-learning controllers generated in simulated locomotion and manipulation environments. It involves no human subjects and no personally identifiable data. The paper describes a failure mode of existing in-context agents and a training-free way to avoid it at deployment. We are not aware of harmful applications that this work enables beyond those already possible with the algorithms we evaluate. We declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

Appendix B specifies every task family, the task parameter that varies within it, and the held-out split. It records how each offline dataset was collected and padded. It also gives the architecture, training schedule, and deployment context rule of every baseline, the evaluation harness and its seeding, and the compute used. Appendix E states every context rule of the ablation exactly, and Appendix A proves every theoretical statement. Appendix H says what is released and how each printed number can be traced to the evaluation records that produced it. The supplementary material contains the code for exploratory prompting and prompt dropout: the context rule for both lineages, the mask schedule, a toy setting with an entry point that runs the rules, and a smoke test.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

Samuel Beaussant and Mehdi Mounsif. Scaling algorithm distillation for continuous control with Mamba. *arXiv preprint arXiv:2506.13892*, 2025. URL https://arxiv.org/abs/2506.13892.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A tutorial on meta-reinforcement learning. *Foundations and Trends in Machine Learning*, 18(2–3):224–384, 2025. doi: 10.1561/2200000080. URL https://doi.org/10.1561/2200000080.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf.

Weiqin Chen and Santiago Paternain. Random policy enables in-context reinforcement learning within trust horizons. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=mAiMKnr9r5.

Weiqin Chen, Xinjie Zhang, Dharmashankar Subramanian, and Santiago Paternain. Filtering learning histories enhances in-context reinforcement learning. *arXiv preprint arXiv:2505.15143*, 2025. URL https://arxiv.org/abs/2505.15143.

Zhenwen Dai, Federico Tomasi, and Sina Ghiassian. In-context exploration-exploitation for reinforcement learning. In *International Conference on Learning Representations*, pp. 47335–47350, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cf3996960b227152edb6afbc073cf322-Abstract-Conference.html.

Juncheng Dong, Moyang Guo, Ethan X. Fang, Zhuoran Yang, and Vahid Tarokh. In-context reinforcement learning from suboptimal historical data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 14021–14039. PMLR, 2025. URL https://proceedings.mlr.press/v267/dong25d.html.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning – identifiability challenges and effective data collection strategies. In *Advances in Neural Information Processing Systems*, volume 34, pp. 4607–4618. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/248024541dbda1d3fd75fe49d1a4df4d-Paper.pdf.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016. URL https://arxiv.org/abs/1611.02779.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025. URL https://arxiv.org/abs/2505.23458.

Yunkai Gao, Rui Zhang, Jiaming Guo, Fan Wu, Qi Yi, Shaohui Peng, Siming Lan, Ruizhi Chen, Zidong Du, Xing Hu, Qi Guo, Ling Li, and Yunji Chen. Context shift reduction for offline meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 80024–80043. Curran Associates, Inc., 2023. doi: 10.52202/075280-3505. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fd489a44f3bcb9f122e4931ef21d0c43-Paper-Conference.pdf.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. URL https://arxiv.org/abs/2207.12598.

Shengchao Hu, Wanru Zhao, Weixiong Lin, Li Shen, Ya Zhang, and Dacheng Tao. Prompt tuning with diffusion for few-shot pre-trained policy generalization. *arXiv preprint arXiv:2411.01168*, 2024. URL https://arxiv.org/abs/2411.01168.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Prompt tuning with preference ranking for few-shot pre-trained decision transformer. *Science China Information Sciences*, 69(1): 112105, 2026. doi: 10.1007/s11432-024-4545-1. URL https://doi.org/10.1007/s11432-024-4545-1.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 34, pp. 1273–1286. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Akshay Krishnamurthy, Keegan Harris, Dylan J. Foster, Cyril Zhang, and Aleksandrs Slivkins. Can large language models explore in-context? In *Advances in Neural Information Processing Systems*, volume 37, pp. 120124–120158. Curran Associates, Inc., 2024. doi: 10.52202/079017-3818. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d951f73c521d069fefbb73396df01424-Abstract-Conference.html.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=hy0a5MMPUv.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. doi: 10.1017/9781108571401.

Jonathan N. Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43057–43083. Curran Associates, Inc., 2023. doi: 10.52202/075280-1866. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8644b61a9bc87bf7844750a015feb600-Paper-Conference.pdf.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. In *International Conference on Learning Representations*, pp. 31176–31233, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/84b5e56a80cbea27de332d456f0d5a4c-Abstract-Conference.html.

Evan Zheran Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6925–6935. PMLR, 2021. URL https://proceedings.mlr.press/v139/liu21s.html.

Jinmei Liu, Fuhong Liu, Zhenhong Sun, Jianye Hao, Huaxiong Li, Bo Wang, Daoyi Dong, Chunlin Chen, and Zhi Wang. Scalable in-context Q-learning. In *International Conference on Learning Representations*, pp. 97519–97539, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/9e76870b97621b3153b84782062143ef-Abstract-Conference.html.

Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, K. R. Zentner, Ryan Julian, Jordan Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Meta-World+: An improved, standardized, RL benchmark. In *Advances in Neural Information Processing Systems*, volume 38, pp. 164667–164687, 2025. doi: 10.52202/085713-4960. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/da295a7037612bb3bfc1334a6f8fc8bd-Abstract-Datasets_and_Benchmarks_Track.html.

Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Dieter Fox, and Abhishek Gupta. ASID: Active exploration for system identification in robotic manipulation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jNR6s6OSBT.

Amir Moeini, Jiuqi Wang, Jacob Beck, Ethan Blaser, Shimon Whiteson, Rohan Chandra, and Shangtong Zhang. A survey of in-context reinforcement learning. *arXiv preprint arXiv:2502.07978*, 2025. URL https://arxiv.org/abs/2502.07978.

Giovanni Monea, Antoine Bosselut, Kianté Brantley, and Yoav Artzi. LLMs are in-context bandit reinforcement learners. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2410.05362.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://www.roboticsproceedings.org/rss20/p090.html.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, volume 26, pp. 3003–3011. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/hash/6a5889bb0190d0211a991f47bb19a777-Abstract.html.

Andrei Polubarov, Nikita Lyubaykin, Alexander Derevyagin, Ilya Zisman, Denis Tarasov, Alexander Nikulin, and Vladislav Kurenkov. Vintix: Action model via in-context reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 49569–49602. PMLR, 2025. URL https://proceedings.mlr.press/v267/polubarov25a.html.

Andrei Polubarov, Nikita Lyubaykin, Alexander Derevyagin, Artyom Grishin, Igor Saprygin, Aleksandr Serkov, Mark Averchenko, Daniil Tikhonov, Maksim Zhdanov, Alexander Nikulin, Ilya Zisman, Albina Klepach, Alexey Zemtsov, and Vladislav Kurenkov. Vintix II: Decision Pre-Trained Transformer is a scalable in-context reinforcement learner. In *International Conference on Learning Representations*, pp. 56944–56997, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d83bdbcdeba05c276c8720a3f97a049-Abstract-Conference.html.

Vitchyr H. Pong, Ashvin V. Nair, Laura M. Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17811–17829. PMLR, 2022. URL https://proceedings.mlr.press/v162/pong22a.html.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

Finn Rietz, Oleg Smirnov, Sara Karimi, and Lele Cao. Prompt tuning decision transformers with structured and scalable bandits. In *Advances in Neural Information Processing Systems*, volume 38, pp. 58258–58286. Curran Associates, Inc., 2025. doi: 10.52202/085713-1953. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/548551c07a68c8f0a87d67c6167cedb1-Abstract-Conference.html.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Thomas Schmied, Fabian Paischer, Vihang Prakash Patil, Markus Hofmarcher, Razvan Pascanu, and Sepp Hochreiter. Retrieval-augmented decision transformer: External memory for in-context RL. In *Proceedings of the 4th Conference on Lifelong Learning Agents*, volume 330 of *Proceedings of Machine Learning Research*, pp. 376–417. PMLR, 2026. URL https://proceedings.mlr.press/v330/schmied26a.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Jaehyeon Son, Soochan Lee, and Gunhee Kim. Distilling reinforcement learning algorithms for in-context model-based planning. In *International Conference on Learning Representations*, pp. 37293–37310, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5c4a7643ab90cd62320df95c873a1c6f-Abstract-Conference.html.

Kefan Song, Amir Moeini, Peng Wang, Lei Gong, Rohan Chandra, Shangtong Zhang, and Yanjun Qi. Reward is enough: LLMs are in-context reinforcement learners. In *International Conference on Learning Representations*, pp. 112747–112770, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b7511dfe2e7a1fa45e093cc75389abc2-Abstract-Conference.html.

Bradly C. Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, pp. 9280–9290. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/d0f5722f11a0cc839fa2ca6ea49d8585-Abstract.html. Indexed in the proceedings as "The Importance of Sampling in Meta-Reinforcement Learning".

Gokul Swamy, Sanjiban Choudhury, J. Andrew Bagnell, and Zhiwei Steven Wu. Sequence model imitation learning with unobserved contexts. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17665–17676. Curran Associates, Inc., 2022. doi: 10.52202/068431-1284. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/708e58b0b99e3e62d42022b4564bad7a-Abstract-Conference.html.

Denis Tarasov, Alexander Nikulin, Ilya Zisman, Albina Klepach, Andrei Polubarov, Nikita Lyubaykin, Alexander Derevyagin, Igor Kiselev, and Vladislav Kurenkov. Yes, Q-learning helps offline in-context RL. *Reinforcement Learning Journal*, 7, 2026. URL https://rlj.cs.umass.edu/2026/papers/Paper3.html.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109.

Zhi Wang, Li Zhang, Wenhao Wu, Yuanheng Zhu, Dongbin Zhao, and Chunlin Chen. Meta-DT: Offline meta-RL as conditional sequence modeling with world model disentanglement. In *Advances in Neural Information Processing Systems*, volume 37, pp. 44845–44870. Curran Associates, Inc., 2024. doi: 10.52202/079017-1425. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4f3820576130a8f796ddbf204c841487-Abstract-Conference.html.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2564–2575. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/1b113258af3968aaf3969ca67e744ff8-Abstract.html.

Wenhao Wu, Fuhong Liu, Haoru Li, Zican Hu, Daoyi Dong, Chunlin Chen, and Zhi Wang. Mixture-of-experts meets in-context reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 24751–24785. Curran Associates, Inc., 2025. doi: 10.52202/085713-0832. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/233cbc888ef01917d8d895e471e80a01-Abstract-Conference.html.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua B. Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *Proceedings of the*

39th International Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pp. 24631–24645. PMLR, 2022. URL https://proceedings.mlr.press/v162/xu22g.html.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Jin Zhang, Jianhao Wang, Hao Hu, Tong Chen, Yingfeng Chen, Changjie Fan, and Chongjie Zhang. MetaCURE: Meta reinforcement learning with empowerment-driven exploration. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12600–12610. PMLR, 2021. URL https://arxiv.org/abs/2006.08170.

Wenhao Zhang, Shao Zhang, Xihuai Wang, Yang Li, and Ying Wen. Towards monotonic improvement in in-context reinforcement learning. *arXiv preprint arXiv:2509.23209*, 2025. URL https://arxiv.org/abs/2509.23209.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27042–27059. PMLR, 2022. URL https://proceedings.mlr.press/v162/zheng22c.html.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A very good method for Bayes-adaptive deep RL via meta-learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkl9JlBYvr.

Luisa Zintgraf, Leo Feng, Cong Lu, Maximilian Igl, Kristian Hartikainen, Katja Hofmann, and Shimon Whiteson. Exploration in approximate hyper-state space for meta reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12991–13001. PMLR, 2021. URL https://proceedings.mlr.press/v139/zintgraf21a.html.

Ilya Zisman, Vladislav Kurenkov, Alexander Nikulin, Viacheslav Sinii, and Sergey Kolesnikov. Emergence of in-context reinforcement learning from noise distillation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62832–62846. PMLR, 2024. URL https://proceedings.mlr.press/v235/zisman24a.html.

A PROOFS

This appendix restates the two lemmas of Section 6 in full, each with a part (iii) that appears only here, states Propositions 3 and 4, and proves them all under Assumptions D, E, G and S, each stated where it is first used. The remarks after the proofs say which assumptions our experiments meet: Lemma 2(iii) rests on one that our data do not meet, and Proposition 3 does not explain the Reach results. The results formalize mechanisms and predict none of our numbers.

Notation. A task $\theta \in \Theta$ is drawn from the prior $\mathcal{P}$. The action space is the normalized box $\mathcal{A} = [-1, 1]^{d_a}$, and the uniform random policy draws every action independently and uniformly from $\mathcal{A}$. An episode $\tau = (s_t, a_t, r_t)_{t=1}^{H}$ records at each step the observation $s_t \in \mathcal{S}$ that the model reads, the action and the reward. The episode is an element of $\mathcal{T} = (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^H$, and $G(\tau) = \sum_{t=1}^{H} r_t$ is its return. A context is a finite sequence of episodes, an element of $\mathcal{C} = \bigcup_{n \geq 0} \mathcal{T}^n$, $\varnothing$ is the empty context, and $\tau_{k,<t} = (s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1})$ is the part of episode k played before step t. Every space carries its Borel σ-algebra, and every set and function below is measurable.

A.1 LEMMA 1: FIXED POINTS OF COLD-START DEPLOYMENT

Assumption D (Deterministic deployment). Fix a task θ. (a) The simulator is deterministic: it has a state x_t with $x_1 = \iota_\theta(\xi_k)$ for a reset draw ξ_k in a set Ξ, $x_{t+1} = f_\theta(x_t, a_t)$ and $r_t = R_\theta(x_t, a_t)$, and the model reads the observation $s_t = h(x_t)$. The reset is deterministic when ι_θ is constant. The state carries a termination flag. Once it is set, the harness freezes the state and returns reward zero (Appendix B). (b) In episode k the action at step t is $a_t = \mu_\phi(s_t, C_k, \tau_{k,<t})$, where C_k is the context at the start of the episode, except that the harness sets $a_t = 0$ after a termination. The DPT lineage ignores the third argument, which Section 3, Algorithm 1 and Appendix E suppress.

Under Assumption D the recursion in t determines the episode from the context C and the reset draw ξ. We write $F_\theta(C; \xi)$ for it, and $F_\theta(C)$ when the reset is deterministic. Deployment is $\tau_k = F_\theta(C_k; \xi_k)$ and $C_{k+1} = U(C_k, \tau_k)$ with $C_1 = \varnothing$, as in (1). The DPT rule is $U(C, \tau) = \tau$. Under the AD rule the context is a buffer of at most M episodes in ascending order of return, with ties broken by a fixed total order on episodes, an idealization that matters only for exact ties between distinct episodes. A sorted buffer is therefore determined by the multiset of its entries. The update $U(C, \tau) = \mathrm{sort}_{G\uparrow}(\mathrm{last}_M(C \oplus \tau))$ appends τ and, if the buffer then holds $M + 1$ episodes, drops the first entry of C, its lowest-return episode. The newest episode always survives, and eviction is decided by return, not by age, as in the released evaluation and our harness (Appendix B). The sorted buffer is the state of (1).

Lemma 1 (restated; part (iii) and the AD-rule clause of part (ii) are stated only here). *Let Assumption D hold.*

(i) Let $\mathfrak{S}$ be a set of contexts such that $U(C, F_\theta(C; \xi)) \in \mathfrak{S}$ for every $C \in \mathfrak{S}$ and every $\xi \in \Xi$. If $C_k \in \mathfrak{S}$ for some $k \geq 1$, then for every realization of the resets and every $j \geq k$, $C_j \in \mathfrak{S}$ and $\tau_j \in F_\theta(\mathfrak{S}; \Xi) = \{F_\theta(C; \xi) : C \in \mathfrak{S},\ \xi \in \Xi\}$. Hence $G(\tau_j)$ lies between the infimum and the supremum of G over $F_\theta(\mathfrak{S}; \Xi)$.

(ii) Let the reset be deterministic. If $U(C^\star, F_\theta(C^\star)) = C^\star$ and $C_k = C^\star$, then $C_j = C^\star$ and $\tau_j = F_\theta(C^\star) = \tau_k$ for every $j \geq k$. In particular, $G(\tau_j) = G(\tau_k)$. Under the DPT rule $C^\star = \tau^\star$ is an episode and the condition reads $F_\theta(\tau^\star) = \tau^\star$. Under the AD rule a fixed point is a full buffer, and the condition reads $F_\theta(C^\star) = C^\star_1$, its lowest-return entry, which the update drops and rewrites. A fixed point need not consist of identical episodes.

(iii) Let the reset be deterministic and the rule the AD rule with buffer size $M \geq 1$. Let $\tau^\star \in \mathcal{T}$ and $\mathcal{W} \subseteq \mathcal{T}$, and suppose $F_\theta(B) = \tau^\star$ for every sorted buffer B of at most M episodes whose entries lie in $\mathcal{W} \cup \{\tau^\star\}$ and include $\tau^\star$. If C_k is such a buffer, then $\tau_j = \tau^\star$ for every $j \geq k$, and for every $j \geq k + M - 1$ the buffer C_j consists of the entries of C_k that follow $\tau^\star$ in the sort order together with copies of $\tau^\star$. This buffer is a fixed point in the sense of (ii), and it is M copies of $\tau^\star$ exactly when no entry of C_k follows $\tau^\star$.

Proof. (i) Induction on j. The case $j = k$ is the hypothesis. If $C_j \in \mathfrak{S}$, then $\tau_j = F_\theta(C_j; \xi_j) \in F_\theta(\mathfrak{S}; \Xi)$ and $C_{j+1} = U(C_j, \tau_j) \in \mathfrak{S}$ by the invariance of $\mathfrak{S}$, whatever ξ_j is. The return bounds follow from $\tau_j \in F_\theta(\mathfrak{S}; \Xi)$.

(ii) Apply (i) to $\mathfrak{S} = \{C^\star\}$. Since the reset is deterministic, $\tau_j = F_\theta(C^\star)$ for every $j \geq k$. Under the DPT rule $U(C^\star, F_\theta(C^\star)) = F_\theta(C^\star)$, so the condition is $F_\theta(C^\star) = C^\star$. Under the AD rule $U(C^\star, \tau)$ has more entries than $C^\star$ unless $C^\star$ is full. For a full $C^\star = (C^\star_1, \ldots, C^\star_M)$ the update is the sorted buffer with entries $C^\star_2, \ldots, C^\star_M, \tau$, which equals $C^\star$ if and only if $\tau = C^\star_1$, because a sorted buffer is determined by the multiset of its entries.

(iii) Let $\mathcal{B}$ be the set of sorted buffers of at most M episodes with entries in $\mathcal{W} \cup \{\tau^\star\}$ that include $\tau^\star$. If $C_j \in \mathcal{B}$, then $\tau_j = F_\theta(C_j) = \tau^\star$ by hypothesis, and $C_{j+1} = U(C_j, \tau^\star)$ has at most M entries, each an entry of C_j or $\tau^\star$, and includes $\tau^\star$. Hence $C_{j+1} \in \mathcal{B}$, and by induction $C_j \in \mathcal{B}$ and $\tau_j = \tau^\star$ for every $j \geq k$, which is (i) with $\mathfrak{S} = \mathcal{B}$. For the limit, call an entry of C_k low if it precedes $\tau^\star$ in the sort order and high if it follows. From episode k on, each update appends $\tau^\star$ and, once the buffer is full, drops its first entry. A low entry, when present, is the first entry. So low entries are dropped before any copy of $\tau^\star$. A high entry is never first, since every C_j with $j \geq k$ contains $\tau^\star$. So high entries are never dropped. The buffer is full after at most $M - |C_k|$ updates and holds at most $|C_k| - 1$ low entries. So after at most $M - 1$ updates none remains. From then

on the buffer consists of the high entries of C_k and copies of $\tau^\star$, its first entry is a copy of $\tau^\star$, and each update drops one copy and appends another. The buffer is then a fixed point by (ii), and holds M copies of $\tau^\star$ exactly when C_k has no high entry. □

Remark 1 (Deterministic and random resets). Part (ii) applies where the reset is deterministic, as in MetaWorld, whose reruns are bit-identical. But no AD- or DPT-lineage trial there reaches an exact fixed point: none of their 4,320 consecutive episode pairs (480 trials) repeats exactly, although the same check finds a repeat in all 1,080 pairs of Prompt-DT with a fixed offline prompt, and late episodes repeat only approximately (44% of the pairs within episodes 6 to 10 differ in return by less than 1%). Under their released rules Meta-DT and S-ICQL repeat exactly in only 1 of 1,080 and 9 of 1,080 pairs. With a random reset an exact fixed point would need the same episode after every reset. So part (i) is the version that applies. On HalfCheetah-Vel, take $\mathfrak{S}$ to be the buffers of episodes that keep the robot in place. If each produces another such episode after every reset, the trial never leaves $\mathfrak{S}$, and every return stays near the standing value $-H\theta$ of (2) up to the control cost, which a robot that does not move can still pay. The standing AD-lineage trials of Section 4, such as the probe trials of Appendix F.1 that return -429 to -422 and -585 to -579 in every episode, are consistent with such a set. So are the stuck AD-lineage HalfCheetah-Vel trials that return within 5% of -200θ in every one of episodes 6 to 10 (86 of 169, or 103 by their last-5 mean; Appendix F.2). The overshooting DPT-lineage chains are not, since their speed keeps rising. Whether a trained model has such an invariant set is an empirical question; the lemma says only what follows once it does.

Remark 2 (Pretraining constrains the model only where it has mass). Let z denote a model input (observation, context and current partial episode) in a separable metric space $\mathcal{Z}_{\text{in}}$, and suppose that pretraining minimizes $L(\mu) = \mathbb{E}_{(z,y)\sim\mathcal{D}_{\text{pre}}}\big[\text{err}(\mu(z), y)\big]$ over a class of predictors, where $\text{err} \geq 0$ is a pointwise loss, y the training target (the next action for AD, the task's action label for DPT), $\mathcal{D}_{\text{pre}}$ here the joint law of input and target and $\mathcal{D}_z$ its input marginal. If $\mu' = \mu$ outside a set A and $\text{err} \leq \bar{b}$, then $|L(\mu') - L(\mu)| \leq \bar{b}\,\mathcal{D}_z(A)$: the objective registers a change on A only in proportion to the pretraining mass of A. The extreme case is the complement O of $\text{supp}(\mathcal{D}_z)$, the inputs with a null open neighborhood; O is open, and it is $\mathcal{D}_z$-null because by separability countably many such neighborhoods cover it. Hence every μ' in the class that agrees with a minimizer on the support is a minimizer, whatever it does on O. The objective therefore does not exclude a fixed point of Lemma 1 all of whose model inputs $(s_t, C^\star, \tau_{<t})$, with $\tau = F_\theta(C^\star)$, lie in O, provided the class contains a predictor that agrees with a minimizer on the support and realizes it. Conversely, two continuous predictors that agree $\mathcal{D}_z$-almost everywhere agree on the whole support, since $\{\mu \neq \mu'\}$ is open and an open set that meets the support has positive probability. Two caveats apply. First, suppose the behavior policy has a full-support action density, such as the tanh-squashed Gaussians of our SAC runs, and the simulator and reset depend continuously on their inputs. Every trajectory reachable with actions in $\mathcal{A}$ from a reset in the support of the reset law then lies in the support of the pretraining trajectory law. Also self-written contexts of a pretraining task that have a pretraining form, a DPT prompt or an AD buffer of at most three episodes, lie in $\text{supp}(\mathcal{D}_z)$, and the objective constrains the model there only through their small pretraining mass. A full AD buffer is read with four earlier episodes, while a pretraining window of four shows the model at most three. So in a metric that separates input lengths those inputs lie in O. Second, this is a statement about the objective, not about the trained network: whether the network realizes such a fixed point is decided by its architecture and training, and Section 4 measures how often it does.

A.2 Lemma 2: coverage by change of measure

Let $(\mathcal{Z}, \mathcal{F})$ be a measurable space and $\mathcal{D}_{\text{pre}}$ and Q probability measures on it. In the application $\mathcal{Z} = \Theta \times \mathcal{C}$ holds a task and a context, $\mathcal{D}_{\text{pre}}$ is its law in pretraining and Q its law in deployment. Wherever open sets or continuity appear, $\mathcal{Z}$ is a separable metric space with metric d and $\mathcal{F}$ its Borel σ-algebra. Expectations of $[0,\infty]$-valued functions may be infinite. The loss of the application is $\ell(\theta, C) = \mathbb{E}\big[\text{err}\big(\mu_\phi(s, C, h), y\big)\,\big|\,\theta, C\big]$, the expected error over a query observation s, current partial episode h and target y drawn as in pretraining given (θ, C).

Assumption E (An early, random-policy-like start in the pretraining contexts). (a) Pretraining and deployment draw the task from the same prior $\mathcal{P}$. (b) There are Markov kernels D_θ, P_θ^{early} and P_θ^{unif} from Θ to $\mathcal{T}$: the law of the first episode of a pretraining context of task θ, the trajectory

law of an early behavior policy in task θ, and the trajectory law of the uniform random policy in task θ. (c) There are constants $\alpha \in (0, 1]$ and $\rho \in [1, \infty)$ such that, for $\mathcal{P}$-almost every θ and every measurable $B \subseteq \mathcal{T}$,

$$D_\theta(B) \;\geq\; \alpha\, P_\theta^{\mathrm{early}}(B) \qquad \text{and} \qquad P_\theta^{\mathrm{unif}}(B) \;\leq\; \rho\, P_\theta^{\mathrm{early}}(B).$$

The first inequality holds when D_θ is a mixture that gives weight at least α to $P_\theta^{\mathrm{early}}$, as for the DPT prompt, one episode drawn uniformly from the task's data, with α the share of the early policy's episodes. The first entry of an AD training window, the lowest-return of four random episodes, is not such a mixture, and the inequality is then a condition on its law. Taking $B = \mathcal{T}$ shows $\rho \geq 1$. Random-level episodes are rare in our data: 0.13% of the HalfCheetah-Vel pretraining episodes (10 of 7,410; none on 10 of the 15 tasks) and 1.6% to 13.1% on the other families return less than a uniformly random episode on the same task, and the first tenth of each HalfCheetah-Vel task's data averages -252, against -444 for a random episode. Throughout, D_θ is the population law of the data-generating process, not the empirical law of the dataset: an empirical measure is carried by finitely many trajectories, to which any atomless deployment law gives no mass, and, by the remark below, neither does the self-episode law. So a finite-sample reading would be vacuous. For the same reason (a) replaces the finite set of training tasks by the prior they are drawn from, as Lin et al. (2024) do; with finitely many training tasks, held-out tasks have pretraining probability zero and part (ii) applies whatever the context. The per-task bound of part (iii) needs only (b) and (c) at the task in question, training tasks included, on which stuck trials also occur (Appendix F.1); (a) only integrates it over tasks.

Lemma 2 (restated and extended). *(i) Suppose $Q(A) \leq \kappa\, \mathcal{D}_{\mathrm{pre}}(A)$ for every $A \in \mathcal{F}$, for some $\kappa < \infty$. Then $\kappa \geq 1$, Q is absolutely continuous with respect to $\mathcal{D}_{\mathrm{pre}}$ with $dQ/d\mathcal{D}_{\mathrm{pre}} \leq \kappa$ $\mathcal{D}_{\mathrm{pre}}$-almost everywhere, and $\mathbb{E}_Q[\ell] \leq \kappa\, \mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell]$ for every measurable $\ell : \mathcal{Z} \to [0, \infty]$. Conversely, absolute continuity with $dQ/d\mathcal{D}_{\mathrm{pre}} \leq \kappa$ almost everywhere implies the setwise condition.*

(ii) If $Q(A) > 0 = \mathcal{D}_{\mathrm{pre}}(A)$ for some $A \in \mathcal{F}$, then for every $c > 0$ the loss $c\,\mathbf{1}_A$ has pretraining risk 0 and deployment risk $c\,Q(A)$. So no $\kappa < \infty$ satisfies $\mathbb{E}_Q[\ell] \leq \kappa\, \mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell]$ for every ℓ, nor for every 1-Lipschitz ℓ with values in $[0, 1]$. If moreover A is open, the loss $\ell_A(z) = \min\{1, \mathrm{d}(z, \mathcal{Z} \setminus A)\}$ is 1-Lipschitz, vanishes outside A, and has pretraining risk 0 and positive deployment risk.

(iii) Under Assumption E(b), fix $\alpha \in (0, 1]$ and $\rho \geq 1$ and let θ be a task at which both inequalities of E(c) hold with these constants for every measurable B. Then $P_\theta^{\mathrm{unif}}(B) \leq (\rho/\alpha)\, D_\theta(B)$ for every measurable $B \subseteq \mathcal{T}$. Under all of Assumption E, let $\mathcal{D}^1_{\mathrm{pre}}$ and Q^{unif} be the laws on $\Theta \times \mathcal{T}$ of (task, first context episode) in pretraining and of (task, uniformly random episode), obtained by integrating D_θ and P_θ^{unif} against $\mathcal{P}$. Then $Q^{\mathrm{unif}}(A) \leq (\rho/\alpha)\, \mathcal{D}^1_{\mathrm{pre}}(A)$ for every measurable $A \subseteq \Theta \times \mathcal{T}$. So (i) holds for this pair with $\kappa = \rho/\alpha$.

Proof. (i) $A = \mathcal{Z}$ gives $1 \leq \kappa$. If $\mathcal{D}_{\mathrm{pre}}(A) = 0$ then $Q(A) = 0$. So Q is absolutely continuous and the density $q = dQ/d\mathcal{D}_{\mathrm{pre}}$ exists. If $E = \{q > \kappa\}$ had positive $\mathcal{D}_{\mathrm{pre}}$-measure, then $Q(E) - \kappa\, \mathcal{D}_{\mathrm{pre}}(E) = \int_E (q - \kappa)\, d\mathcal{D}_{\mathrm{pre}} > 0$ would contradict the hypothesis. So $q \leq \kappa$ almost everywhere, and $\mathbb{E}_Q[\ell] = \int \ell\, q\, d\mathcal{D}_{\mathrm{pre}} \leq \kappa\, \mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell]$ for every measurable $\ell \geq 0$. Conversely, $q \leq \kappa$ almost everywhere gives $Q(A) = \int_A q\, d\mathcal{D}_{\mathrm{pre}} \leq \kappa\, \mathcal{D}_{\mathrm{pre}}(A)$.

(ii) $\mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[c\mathbf{1}_A] = c\, \mathcal{D}_{\mathrm{pre}}(A) = 0$ and $\mathbb{E}_Q[c\mathbf{1}_A] = c\, Q(A) > 0$. A finite Borel measure on a metric space is inner regular with respect to closed sets, so A contains a closed F with $Q(F) > 0 = \mathcal{D}_{\mathrm{pre}}(F)$. The loss $\ell_n(z) = \max\{0, 1/n - \mathrm{d}(z, F)\}$ is 1-Lipschitz, equals $1/n$ on F and vanishes outside $F_n = \{z : \mathrm{d}(z, F) < 1/n\}$, so $\mathbb{E}_Q[\ell_n] \geq Q(F)/n$ and $\mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell_n] \leq \mathcal{D}_{\mathrm{pre}}(F_n)/n$. The sets F_n decrease to the closed set F. So $\mathcal{D}_{\mathrm{pre}}(F_n) \to 0$, and $\mathbb{E}_Q[\ell_n] > \kappa\, \mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell_n]$ as soon as $\mathcal{D}_{\mathrm{pre}}(F_n) < Q(F)/\kappa$. Let A be open. Its complement is nonempty, since $\mathcal{D}_{\mathrm{pre}}(\mathcal{Z} \setminus A) = 1$. So $z \mapsto \mathrm{d}(z, \mathcal{Z} \setminus A)$ is finite and 1-Lipschitz, and so is ℓ_A. It vanishes on $\mathcal{Z} \setminus A$, so $\mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell_A] = 0$. For $z \in A$ some ball of radius $r > 0$ about z lies in A, so $\ell_A(z) \geq \min\{1, r\} > 0$, and a function positive on a set of positive Q-measure has a positive integral, so $\mathbb{E}_Q[\ell_A] > 0$.

(iii) For the fixed θ and any measurable B, the two inequalities give $P_\theta^{\mathrm{unif}}(B) \leq \rho\, P_\theta^{\mathrm{early}}(B) \leq (\rho/\alpha)\, D_\theta(B)$. Under all of Assumption E this holds for $\mathcal{P}$-almost every θ. For a Markov kernel

K and a measurable $A \subseteq \Theta \times \mathcal{T}$ with sections $A_\theta = \{\tau : (\theta, \tau) \in A\}$, the map $\theta \mapsto K_\theta(A_\theta)$ is measurable. Applying the per-task bound to $B = A_\theta$ and integrating against $\mathcal{P}$, on which the exceptional null set has no effect, gives $Q^{\mathrm{unif}}(A) \le (\rho/\alpha)\, \mathcal{D}^1_{\mathrm{pre}}(A)$. □

Remark 3 (How to read Lemma 2). *The first self episode.* Suppose that, conditionally on the task, the checkpoint and the initial observation, the behavior policies that generated the pretraining data give the first action a density with respect to Lebesgue measure on $\mathcal{A}$. The tanh-squashed Gaussian policies of SAC (Haarnoja et al., 2018) do, and so does any mixture of them over checkpoints. With a deterministic head, the first action of the empty-context episode is $\mu_\phi(s_1, \varnothing, \varnothing)$, a function of s_1 alone. Let S be the set of (task, episode) pairs whose first action equals $\mu_\phi(s_1, \varnothing, \varnothing)$ at the episode's own s_1. Every self first episode lies in S. So $Q(S) = 1$ for the law Q of (task, self first episode), pooled over tasks and resets, whereas a pretraining rollout lies in S with probability zero given the task, the checkpoint and s_1. A DPT prompt is one such rollout, and the first AD entry is one of four, so $\mathcal{D}_{\mathrm{pre}}(S) = 0$. Hence Q is singular to $\mathcal{D}_{\mathrm{pre}}$ for every deterministic head, and part (ii) applies to every deterministic self first episode, stuck or not. Later self episodes depend on earlier resets as well, and the argument does not extend to them, except under a deterministic reset, where every self episode is one point per task. At this level the lemma cannot tell a stuck self episode from a harmless one: it says only that no change-of-measure bound holds for every loss.

Continuous losses and the support. The witness $\mathbf{1}_S$ is the indicator of a thin set, but continuity does not rescue a bound: by part (ii) no finite κ serves every 1-Lipschitz loss either. The losses that matter are errors of a network, continuous when the network and a bounded pointwise error are and the law of the query and target varies continuously with (θ, C). For continuous losses the support decides whether a loss can have zero pretraining risk yet positive deployment risk. Let O be the complement of $\mathrm{supp}(\mathcal{D}_{\mathrm{pre}})$, open and $\mathcal{D}_{\mathrm{pre}}$-null as in Remark 2. If $\ell \ge 0$ is continuous with $\mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell] = 0$, then $\ell = 0$ on the support, since otherwise the open set $\{\ell > 0\}$ would meet the support and have positive probability. The deployment risk of such a loss is then $\int_O \ell \, dQ$, zero when $Q(O) = 0$, while for $Q(O) > 0$ the loss ℓ_O of part (ii) has zero pretraining risk and positive deployment risk. So a continuous loss can have zero pretraining risk and positive deployment risk if and only if Q charges O. A tanh-squashed Gaussian gives positive probability to every open set of action sequences. Consequently, under a simulator and reset that depend continuously on their inputs, every self-written episode of a pretraining task, stuck or not, has small but positive pretraining probability in each of its neighborhoods (Remark 2). For our data the dichotomy of part (ii) is then a matter of degree, a large though finite ratio on each fixed neighborhood that grows without bound as the neighborhood shrinks, sharpest for a statistic of the episode that the early policies rarely produce. This is the sense in which Section 4 calls the self-written first episode atypical of the pretraining data and the random episode far from it. If ℓ depends on the context only through a statistic T, part (i) applies to the image laws $T_\# Q$ and $T_\# \mathcal{D}_{\mathrm{pre}}$. Their ratio is at most κ, since $T_\# Q(B) = Q(T^{-1}B) \le \kappa\, T_\# \mathcal{D}_{\mathrm{pre}}(B)$. The ratio can be much smaller. Part (ii) is informative about the attractor only through such a statistic, one whose values on stuck episodes the pretraining episodes rarely or never produce: on HalfCheetah-Vel, speed relative to the target for the DPT lineage and per-step stillness and action variability for the AD lineage (Section 4, Appendix F.4). Such a statistic separates self episodes from the pretraining data, not stuck trials from the others, and a random episode is not typical of pretraining either: its action variability exceeds that of 99% of the pretraining episodes.

Size of ρ. Suppose P^{unif}_θ and $P^{\mathrm{early}}_\theta$ are the trajectory laws of two policies in the same task, with the same reset law and the simulator of Assumption D(a). Suppose too that the early policy has an action density $\pi^{\mathrm{early}}(a \mid s_t, \tau_{<t}) \ge 1/(\rho_0 \,\mathrm{vol}(\mathcal{A}))$ for all arguments. With respect to the product of the reset law and Lebesgue measure on $\mathcal{A}^H$, the joint density of the reset draw and the action sequence is $\mathrm{vol}(\mathcal{A})^{-H}$ under the uniform policy and $\prod_t \pi^{\mathrm{early}}(a_t \mid s_t, \tau_{<t})$ under the early policy, so the first is at most ρ_0^H times the second. Both trajectory laws are images of these laws under the same map from (reset draw, actions) to trajectories, and images preserve setwise inequalities, so $\rho \le \rho_0^H$. The ratio can compound over the $H = 200$ steps of an episode; so the lemma separates a finite ratio from an infinite one rather than predicting a size. For the tanh-squashed Gaussian policies of our SAC runs, the action density vanishes at the faces of $\mathcal{A}$ faster than any power of the distance to the face, while the uniform density does not. So ρ_0 and ρ are infinite. Assumption E(c) fails literally for these data whatever early policy is chosen, since by part (iii) it would give $P^{\mathrm{unif}}_\theta \le (\rho/\alpha) D_\theta$, while D_θ, built from such policies, gives the episodes whose first action lies within δ of a face a mass that vanishes

faster than any power of δ. The assumption can hold for random episodes drawn uniformly from the shrunk box $[-1+\eta, 1-\eta]^{d_a}$, whose density ratio to a tanh-squashed Gaussian with bounded pre-squash mean and log standard deviation is finite, or for a statistic T that ignores actions within η of the faces. The law of the fully uniform episode is then a mixture $Q^{\mathrm{unif}} = (1-\epsilon)Q_1 + \epsilon Q_2$, with Q_1 the law under uniform actions on the shrunk box and $\epsilon = 1-(1-\eta)^{Hd_a}$ the probability that some action coordinate leaves it. If $dQ_1/d\mathcal{D}_{\mathrm{pre}} \le \kappa$ and $0 \le \ell \le \bar{\ell}$, part (i) applied to Q_1 gives $\mathbb{E}_{Q^{\mathrm{unif}}}[\ell] \le \kappa\, \mathbb{E}_{\mathcal{D}_{\mathrm{pre}}}[\ell] + \epsilon\bar{\ell}$. The mass ϵ is small only for η well below $1/(Hd_a)$, where κ is astronomically large. We do not measure ρ_0, ρ or α and claim no value for them.

What the lemma does and does not give. Our data do not meet Assumption E. So part (iii) is a statement about a pretraining distribution we did not have. Besides the infinite ρ, random-level episodes are rare (see Assumption E). And a uniformly random HalfCheetah-Vel episode is far from the pretraining data step by step (96.4% of its steps lie beyond the 95th-percentile nearest-neighbor distance of held-out-task data, against 13.6% for the earliest tenth of those tasks' own SAC data; Figure 17b). Part (iii) states what a pretraining set would have to contain for one exploratory episode to fall under an existing guarantee. Our frozen models read a random episode anyway (Figure 3), and coverage does not explain why. The lemma bounds the average of a loss on the first context. Turning a small imitation error into a return guarantee needs a further argument of the kind given by Lin et al. (2024), which we do not reproduce. The loss averages over query observations drawn as in pretraining, whereas those met at deployment are the model's own, a shift not covered by κ. Under the AD rule the later entries of the buffer are the agent's own episodes and are not covered.

A.3 Proposition 3: goal identifiability from one episode

Assumption G (The goal enters through distances). The task parameter is a goal $g \in \mathbb{R}^n$ with true value $g^\star$; $n = 3$ in MetaWorld. (a) The goal is hidden from the policy, and the reset law and the dynamics do not depend on it. (b) There are points $x_t \in \mathbb{R}^n$ and quantities o_t, each a function of the recorded observations, and a known function χ such that $r_t = \chi(\|x_t - g^\star\|, o_t)$ for every t, where $d \mapsto \chi(d, o)$ is injective on $[0, \infty)$ for every o.

Given an episode τ, let d_t be the unique $d \ge 0$ with $\chi(d, o_t) = r_t$. By injectivity $d_t = \|x_t - g^\star\|$. The set of goals consistent with τ is $\mathcal{G}(\tau) = \{g \in \mathbb{R}^n : \chi(\|x_t - g\|, o_t) = r_t \text{ for } t = 1, \dots, H\} = \{g : \|x_t - g\| = d_t \text{ for } t = 1, \dots, H\}$, and it contains $g^\star$. If the policy that generated τ depends on g only through the rewards recorded in τ, as the uniform random policy and the empty-context episode 1 of either lineage do, then by (a) the likelihood of τ depends on g only through its rewards. Hence $\mathcal{G}(\tau)$ is exactly the set of goals that could have produced τ, and any posterior over goals given τ alone is supported on it. An episode generated under a goal-informed context, such as an oracle demonstration or a self episode whose prompt carried earlier rewards, can carry further information about g through its actions.

Proposition 3 (Goal identifiability from one episode)**.** *Let Assumption G hold, let τ be an episode with points $x_1, \dots, x_H$, let $V = \operatorname{span}\{x_t - x_1 : 1 \le t \le H\}$, so that $x_1 + V$ is the affine hull of the points and $k = \dim V$, let c be the orthogonal projection of $g^\star$ onto $x_1 + V$, and let $r = \|g^\star - c\|$.*

(i) $g \in \mathcal{G}(\tau)$ if and only if $\|x_1 - g\| = d_1$ and

$$2(x_t - x_1)^\top g = \|x_t\|^2 - \|x_1\|^2 - d_t^2 + d_1^2 \qquad \text{for } t = 1, \dots, H, \tag{3}$$

and $\mathcal{G}(\tau) = \{g \in c + V^\perp : \|g - c\| = r\}$, a sphere of radius r in the affine subspace $c + V^\perp$ of dimension $n - k$.

(ii) For $n = 3$: if $k = 3$, $\mathcal{G}(\tau) = \{g^\star\}$; if $k = 2$, $\mathcal{G}(\tau) = \{c + r\nu, c - r\nu\}$ for a unit normal ν of the plane of the points; if $k = 1$, $\mathcal{G}(\tau)$ is the circle of radius r about c in the plane through c orthogonal to the line of the points; if $k = 0$, $\mathcal{G}(\tau)$ is the sphere of radius d_1 about x_1.

(iii) $\mathcal{G}(\tau) = \{g^\star\}$ if and only if $k = n$ or $g^\star \in x_1 + V$.

(iv) For $n = 3$, if for some indices $t_1, \dots, t_4$ the joint law of $(x_{t_1}, \dots, x_{t_4})$ has a density with respect to Lebesgue measure on $\mathbb{R}^{12}$, then $k = 3$ almost surely.

Proof. (i) For every g and t, $\|x_t - g\|^2 - \|x_1 - g\|^2 = \|x_t\|^2 - \|x_1\|^2 - 2(x_t - x_1)^\top g$. If $g \in \mathcal{G}(\tau)$ the left side is $d_t^2 - d_1^2$, which rearranges to (3). Conversely, if $\|x_1 - g\| = d_1$ and (3) holds, the

identity gives $\|x_t - g\|^2 = d_t^2$ for every t, and both sides are nonnegative. The rows $2(x_t - x_1)^\top$ of the linear system (3) span V, so its null space is $V^\perp$, and $g^\star$ solves it. Its solution set is therefore $g^\star + V^\perp$. Write $g^\star - x_1 = p + q$ with $p \in V$ and $q \in V^\perp$. Then $c = x_1 + p$, $r = \|q\|$ and $g^\star = c + q$. Since $q \in V^\perp$, $g \in g^\star + V^\perp$ if and only if $g - c \in V^\perp$, and then $p \perp g - c$, giving $\|g - x_1\|^2 = \|p\|^2 + \|g - c\|^2$ while $d_1^2 = \|g^\star - x_1\|^2 = \|p\|^2 + r^2$. Hence $g \in \mathcal{G}(\tau)$ if and only if $g - c \in V^\perp$ and $\|g - c\| = r$.

(ii) Here $\dim V^\perp = 3 - k$. If $k = 3$, $V^\perp = \{0\}$ and $c = g^\star$. If $k = 2$, $V^\perp = \mathbb{R}\nu$ and $\{c + s\nu : |s| = r\} = \{c + r\nu, c - r\nu\}$. If $k = 1$, $c + V^\perp$ is the plane through c orthogonal to the line of the points, and the set is the circle of radius r about c in it. If $k = 0$, all points equal x_1, so $c = x_1$ and $r = d_1$.

(iii) If $k = n$ then $c = g^\star$, and if $g^\star \in x_1 + V$ then $r = 0$; in both cases the sphere is $\{g^\star\}$. If $k < n$ and $g^\star \notin x_1 + V$, then $r > 0$ and $V^\perp$ contains a unit vector e, giving two distinct points $c + re$ and $c - re$ of $\mathcal{G}(\tau)$.

(iv) The vectors $x_{t_j} - x_{t_1}$, $j = 2, 3, 4$, lie in V, so $k = 3$ whenever they are linearly independent, that is, whenever $\det[x_{t_2} - x_{t_1},\, x_{t_3} - x_{t_1},\, x_{t_4} - x_{t_1}] \neq 0$. This determinant is a polynomial in the twelve coordinates that is not identically zero, since it equals 1 at $x_{t_1} = 0$ and $x_{t_j} = e_{j-1}$. The zero set of such a polynomial has Lebesgue measure zero. So under a density the event $k < 3$ has probability zero. □

Remark 4 (What the proposition does and does not cover). *Stability.* When $k = n$, the matrix A with rows $2(x_t - x_1)^\top$ has full column rank and $g^\star = A^+ b$, with A^+ the Moore-Penrose pseudoinverse and b the right side of (3). A perturbation δb of the distances moves the least-squares solution by $A^+ \delta b$, of norm at most $\|\delta b\| / s_{\min}(A)$, and the smallest singular value $s_{\min}(A)$ tends to zero as the points approach a common plane. A self Reach episode is not a single point: it moves from the fixed initial hand position to where it hovers, so its points generically span $\mathbb{R}^3$ too, and in this noiseless idealization it also identifies the goal. Conditioning separates the two: the median $s_{\min}(A)$ is 0.088 for self first episodes (78 distinct episodes; an rms spread of 3.1 mm along the thinnest direction) against 0.369 for random episodes (60; 13.1 mm). Under the real reward this does not make the random episode the more informative one.

MetaWorld. Assumption G idealizes the default (V2) rewards of MetaWorld v3 (Yu et al., 2020; McLean et al., 2025). The reach reward is a tolerance function of the gripper-to-goal distance. It is at its maximum within a small radius of the goal, where injectivity fails. Outside that radius it decreases with the gripper-to-goal distance at a scale set by the distance from the fixed initial hand position to the goal, itself unknown to the agent. In Push and Pick-Place the goal enters through the object-to-goal distance. So x_t is the object position, which moves only when the gripper touches the object. An episode that leaves the object in place constrains the goal, through this term, only to a sphere about the object, and the rewards also contain goal-independent terms. The proposition therefore predicts no identifiability gain from a random Push or Pick-Place episode that does not move the object. And random episodes rarely move it (3 of the 60 distinct random episodes on each of Push, Pick-Place and Soccer, judged by a change of more than 2 mm in the minimum or final object-to-goal distance relative to episodes that never touch the object). The proposition also does not explain the Pick-Place gain in success@0.10 (Section 7). On Reach the reward is flat near the fixed start, where a random episode stays (median gripper-to-goal distance 0.26 m over the episode, against 0.13 m for self first episodes). Thus the random episode's rewards constrain the goal less than those of the self episode. The set of goals in MetaWorld's goal space that reproduce all 200 logged rewards of an episode (tolerance: the misfit at the true goal plus 0.05) has a median volume of 1.2 cm^3 for self first episodes (150 episode-goal pairs) against 35 cm^3 for random episodes (60 pairs). The random volume is the larger in 95% of (random, self) pairs. Under Assumption G with distances known within 1 cm, by contrast, the two are comparable (25 and 31 cm^3; not shown). Better conditioning does not give better identification, and Proposition 3 does not explain the Reach results. The self context identifies the goal better than a random one, and the frozen model uses it only in part (Appendix D).

HalfCheetah-Vel. Here the task is a target speed $\theta > 0$ and $r_t = -|v_t - \theta| - c_{\text{ctrl}}\|a_t\|^2$. Suppose the forward velocity v_t that enters the reward can be read from the recorded observations. One step gives $|v_t - \theta| = e_t := -r_t - c_{\text{ctrl}}\|a_t\|^2$, so $\theta \in \{v_t - e_t,\, v_t + e_t\}$. With a training band whose smallest target is $\theta_{\min} > 0$, any step with $v_t < \theta_{\min}$ identifies θ, since the candidate $v_t - e_t \leq v_t$

lies below the band, and an episode whose mean forward speed is below $\theta_{\min}$ contains such a step. The standing episodes of Section 4, with mean speed below 0.1 m/s against a band that starts at 1.1 m/s, meet this condition, and so do the first steps of every episode, since episodes start at rest. The HalfCheetah attractor is not a failure of identifiability: the context contains the task, and the model does not use it.

A.4 Proposition 4: selecting the prompt by the return it induces

Assumption S (Two candidates with sub-Gaussian test returns). Everything below is conditional on the two episodes that define the candidates, the agent's first episode τ_1 and the exploratory episode τ_3. In particular, $G(\tau_3)$ is a fixed number. Candidate $i \in \{1, 2\}$ is the prompt τ_1 or τ_3, with mean induced return $m_i \in \mathbb{R}$, and $\sigma \geq 0$ is a noise level. The two test episodes have returns $Y_i = m_i + \varepsilon_i$. The rule commits to $\hat{\imath} = 2$ if $Y_2 \geq Y_1$ and to $\hat{\imath} = 1$ otherwise. The idealized rule then plays N further episodes with candidate $\hat{\imath}$, with returns $G_j = m_{\hat{\imath}} + \varepsilon'_{\hat{\imath},j}$ for $j = 1, \ldots, N$. The variables ε_1, ε_2 and $\varepsilon'_{i,j}$ are independent, integrable and centered, and $\varepsilon_1, \varepsilon_2$ are σ-sub-Gaussian: $\mathbb{E}\, e^{\lambda \varepsilon_i} \leq e^{\lambda^2 \sigma^2 / 2}$ for every $\lambda \in \mathbb{R}$. For $\sigma = 0$ this forces $\varepsilon_i = 0$ almost surely, since Jensen's inequality gives $\mathbb{E}\, e^{\lambda \varepsilon_i} \geq 1$ with equality only for a constant. Conditioning on (τ_1, τ_3) makes the means well defined, and, given the candidates, the test and committed noises are mutually independent when resets are drawn independently. Averages over the candidates are taken only at the end of part (iii), where $m_{\mathrm{rand}} = \mathbb{E}[G(\tau_3)]$ is the mean return of a uniformly random episode.

Proposition 4 (Selecting the prompt by the return it induces)**.** *Under Assumption S, let* $m^\star = \max(m_1, m_2)$, $\Delta = |m_1 - m_2|$, *and, for the idealized rule, let* $R_N = N m^\star - \mathbb{E}\big[\sum_{j=1}^{N} G_j\big]$ *be the expected shortfall of the committed phase.*

(i) If $\Delta = 0$ *or* $\sigma = 0$, *then* $m_{\hat{\imath}} = m^\star$ *almost surely. If* $\sigma > 0$, *then* $\Pr(m_{\hat{\imath}} < m^\star) \leq e^{-\Delta^2/(4\sigma^2)}$. *This part concerns only the comparison and holds for Algorithm 1 whichever candidate wins.*

(ii) $R_N = N\Delta \Pr(m_{\hat{\imath}} < m^\star)$. *Hence* $R_N = 0$ *if* $\Delta = 0$ *or* $\sigma = 0$, *and* $R_N \leq N\Delta\, e^{-\Delta^2/(4\sigma^2)} \leq N\sigma\sqrt{2/e}$ *if* $\sigma > 0$.

(iii) For the idealized rule on the schedule of Algorithm 1, the expected shortfall over episodes 2 to K, *relative to* $m^\star$ *per episode, equals* $(m^\star - G(\tau_3)) + \Delta + R_N$ *with* $N = K - 4$. *Episode 1 is common to all rules and is not counted. If returns are bounded, averaging over* (τ_1, τ_3) *gives the unconditional shortfall* $\mathbb{E}[m^\star] - m_{\mathrm{rand}} + \mathbb{E}[\Delta + R_N]$.

Proof. (i) If $\Delta = 0$ there is nothing to show. If $\sigma = 0$ and $\Delta > 0$, then $Y_i = m_i$ almost surely and the rule picks the larger mean. Let $\sigma > 0$ and $\Delta > 0$. If $m_1 > m_2$, the event $\{m_{\hat{\imath}} < m^\star\}$ is $\{Y_2 \geq Y_1\} = \{\varepsilon_2 - \varepsilon_1 \geq \Delta\}$; if $m_2 > m_1$, it is $\{Y_2 < Y_1\} \subseteq \{\varepsilon_1 - \varepsilon_2 \geq \Delta\}$. By independence, $\mathbb{E}\, e^{\lambda(\varepsilon_2 - \varepsilon_1)} = \mathbb{E}\, e^{\lambda \varepsilon_2}\, \mathbb{E}\, e^{-\lambda \varepsilon_1} \leq e^{\lambda^2 \sigma^2}$ for every λ. Markov's inequality applied to $e^{\lambda(\varepsilon_2 - \varepsilon_1)}$ with $\lambda = \Delta/(2\sigma^2)$, and symmetrically for $\varepsilon_1 - \varepsilon_2$, then bounds both by $e^{-\Delta^2/(4\sigma^2)}$ (Lattimore & Szepesvári, 2020, Chapter 5). The event depends only on the two test returns, which Algorithm 1 compares in the same way whatever it does afterward.

(ii) $\hat{\imath}$ is a function of $(\varepsilon_1, \varepsilon_2)$, which is independent of every $\varepsilon'_{i,j}$, so $\mathbb{E}[\varepsilon'_{\hat{\imath},j}] = \sum_i \Pr(\hat{\imath} = i)\, \mathbb{E}[\varepsilon'_{i,j}] = 0$ and $R_N = N\big(m^\star - \mathbb{E}[m_{\hat{\imath}}]\big)$. Since $m^\star - m_{\hat{\imath}}$ equals Δ on $\{m_{\hat{\imath}} < m^\star\}$ and 0 otherwise, $R_N = N\Delta \Pr(m_{\hat{\imath}} < m^\star)$. Part (i) gives the two cases and the first inequality. For the second, $f(\Delta) = \Delta\, e^{-\Delta^2/(4\sigma^2)}$ has derivative $e^{-\Delta^2/(4\sigma^2)}\big(1 - \Delta^2/(2\sigma^2)\big)$, so it is maximized at $\Delta = \sqrt{2}\,\sigma$, where $f = \sigma\sqrt{2/e}$.

(iii) Given (τ_1, τ_3), the tests in episodes 2 and 4 have expected returns m_1 and m_2 and fall short of $2m^\star$ by Δ. The exploratory episode 3 contributes $m^\star - G(\tau_3)$, and the committed phase contributes R_N. If returns are bounded, all terms are bounded functions of (τ_1, τ_3), and taking expectations replaces $G(\tau_3)$ by m_{rand} and the other terms by their means. □

Remark 5 (How Proposition 4 maps onto Algorithm 1). In Algorithm 1, candidate 1 is the agent's first episode τ_1 used as the prompt, whose induced return $G(\tau_2)$ is observed in episode 2. The exploratory episode is episode 3, and candidate 2 is that episode used as the prompt, tested in episode

4. The rule selects candidate 2 on a tie, and $N = K - 4 = 6$ episodes remain. The comparison, and hence part (i), is that of Algorithm 1 whichever candidate wins. On the event that candidate 2 wins, its prompt is fixed for episodes 5 to 10, and Algorithm 1 plays exactly the idealized rule of parts (ii) and (iii). Under Assumption D with independent resets, the committed returns are, given (τ_1, τ_3), independent of the tests and have the same mean m_2 as the test return. With the deterministic MetaWorld reset every return is a deterministic function of the prompt, so $\sigma = 0$ and the rule commits to the better prompt exactly. In MuJoCo, a return confined to an interval $[a, b]$ has centered noise that is $\big((b-a)/2\big)$-sub-Gaussian by Hoeffding's lemma (Lattimore & Szepesvári, 2020, Chapter 5); a finite σ therefore exists, but with that value the bound of part (ii) is uninformative. As a scale for σ, the six returns of a committed trial under its fixed prompt (episodes 5 to 10) have a root-mean-square standard deviation of 3.4 on HalfCheetah-Vel, 16 and 20 on Hopper-Friction and Hopper-Mass, 41 and 79 on Walker-Friction and Walker-Mass, and 161 on Humanoid-Dir. The six returns are identical in every MetaWorld trial, as $\sigma = 0$ requires. The standard deviation of a σ-sub-Gaussian variable is at most σ, so these values, which estimate it, are optimistic. In these units the two test returns of a HalfCheetah-Vel trial differ by a median of 8 (T2MIR-DPT) and 14 (DPT) standard deviations, whereas on the other MuJoCo families 71% to 84% of trials have test returns within two standard deviations of each other, including families on which EP wins.

If candidate 1 wins, the rule resumes the rolling self chain from episode 2 instead of fixing τ_1. The prompts of episodes 5 to 10 are then τ_2 and new self episodes; their returns need not have mean m_1, and $G(\tau_2)$ is only a proxy for the future of the self branch. Parts (ii) and (iii) then describe the idealized rule, not Algorithm 1. `always3` commits to the exploratory prompt in 84% of HalfCheetah-Vel trials, including every self-stuck one, and in 33% (Soccer) to 57% (Humanoid-Dir) of the trials of the other families. The losses on Walker-Mass come from the committed branch: in the committed trials the self chain would still have gained 130 to 150 after episode 2, meaning the random prompt wins the comparison with $G(\tau_2)$ yet ends below what the self chain reaches. The proposition says nothing about which candidate is better. That is an empirical matter, and Figure 6 compares the rule with variants that drop either the exploration or the selection. The proposition is the two-armed, one-test-per-arm case of the explore-then-commit analysis of Lattimore & Szepesvári (2020, Chapter 6). Rietz et al. (2025) treat the return induced by a prompt as a bandit reward to tune prompts built from demonstrations, whereas here the candidates are the agent's own episodes and no demonstration is used.

B Setup details

This appendix specifies the setup that the main text takes as given. It covers what a task is in each family and which tasks are held out, and how the offline data were produced. It also covers how each baseline was built and trained and which context rule it follows at deployment, as well as what the evaluation harness shares across methods and what it deliberately leaves to each method, and what the experiments cost.

B.1 Families and held-out tasks

Each family has 20 tasks that share one state space, one action space and a horizon of 200 steps, and differ in the task parameter θ. On HalfCheetah-Vel θ is the target forward speed, with a task band of 1.1 to 3.0 m/s in steps of 0.1. On the four dynamics families it is a multiplicative scale on body mass or on friction. On Humanoid-Dir it is a target heading in radians. On the four MetaWorld families it is a three-dimensional goal position, hidden from the agent: the goal coordinates are removed from MetaWorld's 39-dimensional observation, which leaves 36 dimensions. Reward (the default V2 reward functions) and success come from MetaWorld v3 (Yu et al., 2020; McLean et al., 2025). Of the 20 tasks, 15 are used for training and 5 are held out, one from each block of four consecutive task positions. The six MuJoCo families have an ordered task parameter. In those families, four of the five held-out tasks lie between two training tasks, and the fifth has the largest parameter in the family, one grid step above the largest training parameter (HalfCheetah-Vel 3.0 against 2.9 m/s, the four dynamics families 1.484 against 1.424). Table 3 lists the families and Table 4 the held-out MetaWorld goals.

Table 3: The ten families. $|S|$ and $|A|$ are the observation and action dimensions seen by every method, the episode count is per task (each episode has 200 steps), and the last column lists the parameters of the five held-out tasks. The MetaWorld goals are listed in Table 4.

Family	task parameter θ	$\lvert S\rvert$	$\lvert A\rvert$	episodes per task	held-out θ
MuJoCo locomotion					
HalfCheetah-Vel	target forward speed (m/s)	20	6	494	1.2, 1.5, 2.0, 2.5, 3.0
Hopper-Mass	body mass scale	11	3	494	0.702, 0.795, 0.979, 1.206, 1.484
Hopper-Friction	friction scale	11	3	486	0.702, 0.795, 0.979, 1.206, 1.484
Walker-Mass	body mass scale	17	6	486	0.702, 0.795, 0.979, 1.206, 1.484
Walker-Friction	friction scale	17	6	486	0.702, 0.795, 0.979, 1.206, 1.484
Humanoid-Dir	target heading (rad)	376	17	494	1.728, 2.199, 2.985, 3.770, 4.555
MetaWorld manipulation, goal hidden					
Reach	goal position (x, y, z)	36	4	486	Table 4
Push	goal position (x, y, z)	36	4	412	Table 4
Pick-Place	goal position (x, y, z)	36	4	420	Table 4
Soccer	goal position (x, y, z)	36	4	459	Table 4

Table 4: Held-out goal positions (x, y, z) in meters for the four MetaWorld families, one per block of four consecutive task positions.

Family	task 1	task 2	task 3	task 4	task 5
Reach	$(0.090, 0.800, 0.125)$	$(-0.095, 0.804, 0.053)$	$(0.100, 0.832, 0.091)$	$(-0.098, 0.837, 0.087)$	$(-0.059, 0.886, 0.085)$
Push	$(-0.100, 0.898, 0.015)$	$(-0.098, 0.801, 0.015)$	$(-0.063, 0.874, 0.015)$	$(-0.088, 0.899, 0.015)$	$(-0.050, 0.827, 0.015)$
Pick-Place	$(0.090, 0.800, 0.090)$	$(-0.095, 0.804, 0.052)$	$(0.100, 0.832, 0.072)$	$(-0.098, 0.837, 0.070)$	$(-0.059, 0.886, 0.069)$
Soccer	$(-0.100, 0.898, 0.000)$	$(-0.099, 0.801, 0.000)$	$(-0.067, 0.874, 0.000)$	$(-0.089, 0.899, 0.000)$	$(-0.055, 0.827, 0.000)$

B.2 Offline data

Every task has one SAC run (Haarnoja et al., 2018). The stochastic evaluation rollouts of that run, recorded at many checkpoints, are written in checkpoint order, so a task's data run from the poor behavior of an early checkpoint to the behavior of a trained one. Neither lineage reads this order in training: the AD lineage draws four episodes of one task at random and sorts them by return, and the DPT lineage draws its prompt episode uniformly from the task's data (below). Every pretraining context can therefore contain early, poor episodes, but random-level ones are rare. On HalfCheetah-Vel, 0.13% of the pretraining episodes (10 of 7,410) return less than a uniformly random episode on the same task, and the share is 1.6% to 13.1% on the other families (Appendix A). Within a family every task has the same number of episodes, an evenly spaced subsample of its learning curve (Table 3). Episodes that terminate early are padded to 200 steps with an absorbing state: the last observation is frozen, and the action and reward are zero. The harness pads in the same way at test time, so no method sees a padding convention at deployment that it did not see in training.

On MetaWorld we checked that the offline data contain successful behavior. Counting an episode as successful when it collects MetaWorld's success reward at any step, 80% of episodes succeed on Reach, 72% on Push, 72% on Pick-Place and 73% on Soccer. Low success of a trained model on these families is therefore not a property of the data.

The DPT lineage is trained to predict, at a query state, the action of a good policy for the task (Lee et al., 2023). We label query states with the deterministic action of the task's final SAC actor when that actor is healthy. In families where any of the 20 final actors had collapsed late in training (Hopper-Mass, Hopper-Friction, Walker-Mass, Walker-Friction and Reach), the whole family instead uses the recorded actions of its top-decile offline episodes as labels. We decide per family rather than per task, so that the label source never carries information about which task the model is looking at.

B.3 Baselines and their context rules

Table 5 summarizes the seven methods. Each is trained with the model, dataset and loss code of its official release, vendored without modification. Only the environment plumbing is replaced, so that every method trains on the same offline data and is scored in the same environments.

T2MIR-AD and AD. T2MIR-AD (Wu et al., 2025) is trained on windows of four episodes of one task. The release sorts each task's episodes by return once at load time and draws four of

Table 5: The seven methods and the oracle-demo reference: architecture, training schedule, and the context each one reads at deployment in our harness. "Episode 1" is what the method conditions on at the cold start.

Method	architecture	training	deployment context	episode 1
T2MIR-AD	causal transformer, 3 layers, width 256, 8 heads, mixture-of-experts token and task routing	100k steps, batch 24, lr 3×10^{-4}, 4 episodes of one task drawn at random, sorted by return	buffer of 4 own episodes (800 steps) sorted by return, lowest first; a full buffer drops the lowest-return episode of the previous buffer, and the newest always enters; read at every step	empty
AD	T2MIR-AD without the mixture-of-experts layers	as T2MIR-AD	as T2MIR-AD	empty
T2MIR-DPT	transformer, 4 layers, width 128, 4 heads, mixture-of-experts routing	100k steps, batch 32, prompt: one episode drawn uniformly	previous own episode (200 steps), frozen during the episode, not sorted	empty
DPT	T2MIR-DPT without the mixture-of-experts layers	as T2MIR-DPT	as T2MIR-DPT	empty
Prompt-DT	decision transformer, 3 layers, width 128, context length 20	release trainer, 5-step prompt	5-step segment of the previous own episode; within-episode history reset each episode	empty
Meta-DT	world-model context encoder and decision transformer, horizon 20, width 128, 3 layers	40 epochs, 3.8 to 4.6×10^5 steps (release: 1 to 5×10^5 steps)	per-step code over recent transitions of the current episode; a 5-step prompt from one of the agent's three highest-return earlier episodes, chosen by world-model error	empty
S-ICQL	context encoder and in-context IQL head, 4 layers	400k steps, batch 128	previous own episode, re-encoded into per-step codes before each episode	empty
Prompt-DT (oracle demo)	as Prompt-DT	as Prompt-DT	5-step segment of a top-decile offline episode of the held-out task	demonstration

them without replacement, keeping them in ascending order of return, so the checkpoint order of the data is never seen. At deployment, its released evaluation keeps a buffer of four episodes, re-sorted by return in ascending order after every episode, so that the context looks like an improving history. When a fifth episode ends, the code rolls the buffer by one whole episode before the re-sort; Algorithm 2 of Wu et al. (2025) lists the sort first. The roll drops the lowest-return episode of the previous buffer and keeps the newest one. The model reads this buffer, followed by the current episode, at every step. Algorithm Distillation (Laskin et al., 2023) has no official code release and no continuous-control tasks, so we build AD from the same T2MIR code with the list of mixture-of-experts layers set to empty, a configuration the release supports. Every other setting is shared, including the trainer, the tokenization, the normalization and the context rule. On HalfCheetah-Vel, AD has 2.83M parameters, the number T2MIR-AD uses per token (4.81M in total, of which a token passes through 2.83M via its top-two experts). The two are therefore matched in active, not total, parameters.

T2MIR-DPT and DPT. T2MIR-DPT is trained on pairs of a prompt and a query state. The prompt is one full episode drawn uniformly from the task's offline data, and the query state is labeled as described above. Its released evaluation reads the prompt once at the episode boundary and keeps it fixed for the whole episode. The prompt is always a whole episode and is never sorted, and on the first episode it is empty. The official DPT release (Lee et al., 2023) covers only discrete bandit and navigation environments; DPT is thus built from T2MIR-DPT in the same way as AD. The release's navigation runs use a context of one episode, so their online evaluation also conditions each episode on the previous one. DPT has 0.97M parameters, the number T2MIR-DPT uses per token (1.47M in total).

Prompt-DT. Prompt-DT (Xu et al., 2022) prepends a short trajectory prompt to a decision-transformer stream (Chen et al., 2021) and resets its within-episode history every episode, so any adaptation across episodes has to pass through the prompt. In the released protocol the prompt comes from the evaluation task's own offline demonstrations, which would give Prompt-DT privileged information about a held-out task. Our self-context variant therefore builds the prompt from the agent's own previous episode, cut at the same position as the training prompts, and runs the first episode without a prompt. The oracle-demo reference imitates the released prompt source, the evaluation task's expert demonstrations: a 5-step segment of an episode drawn from the top decile, by return, of the held-out task's offline data. The draw is seeded by the task, so that reruns are identical.

The return-to-go target of Prompt-DT and Meta-DT is derived from the offline returns of the training tasks only.

Meta-DT. Meta-DT (Wang et al., 2024) pairs a context encoder, pretrained with a world-model objective, with a decision transformer. At deployment, its released loop recomputes a context code at every step from the most recent transitions of the current episode and appends it to the sequence the transformer reads. It also fills a separate prompt slot from the agent's own earlier episodes on the task. It takes one of the three highest-return ones at random, scores it with the world model, and draws the start of a 5-step segment with probability proportional to the world-model error from that point on. Our adapter follows this rule, and the first episode has no prompt. The model is trained with the release's default schedule, which adds the prompt after 20k warm-up steps. The release trains the transformer for a fixed number of steps at batch 128 (2×10^5 on HalfCheetah-Vel, 1 to 5×10^5 across its configurations). We train it for 40 epochs at batch 128, which is 3.8 to 4.6×10^5 steps on our data; the training loss flattens between epochs 20 and 40. Meta-DT's context encoder is trained separately with the release's own encoder, decoder and targets.

S-ICQL. S-ICQL (Liu et al., 2026) pairs a context encoder with an in-context implicit Q-learning (IQL) head (Kostrikov et al., 2022). Its released deployment loop collects whole episodes and, before each new one, re-encodes the previous episode into a per-step code sequence, with windows that include the current step. The first episode reads one code computed from an all-zero window. Our adapter follows this loop, and a test that replays the same episodes through both gives identical codes. S-ICQL's encoder is related to Meta-DT's but not identical: its layers are named differently, and its state decoder predicts the state difference rather than the next state. We therefore train the encoder separately, from its own release.

A method we dropped. We also integrated IC-IQL (Tarasov et al., 2026), whose evaluation reads a sliding window of recent steps that is not reset at episode boundaries, but left it out of the grid: we could not validate our training runs of its continuous-control variant.

Why the context rules are not unified. The seven methods read their history in at least five different ways: a return-sorted episode buffer, a prompt frozen at the episode boundary, a short trajectory segment with a history that resets every episode, a code recomputed at every step next to a prompt drawn from the agent's best episodes, and the previous episode re-encoded before each new one. Each model was trained on one of these layouts, and forcing a single layout on all of them would put most of the models out of distribution. The harness therefore shares everything else and leaves the context rule to each release.

B.4 Evaluation harness

Across methods, the harness shares the environment, the cold start, the number of episodes, the episode length, the absorbing-state padding, the return and success bookkeeping, and the environment seed. The environment is built at the absolute task identifier recorded with the dataset and checked against the held-out parameters. For task index i and trial j the environment is seeded with $20260905 + 1000\,i + j$. The seed is passed to the inner MuJoCo environment, because the outer wrapper's seed function does nothing. Every method therefore faces the same initial states in the same order. MuJoCo families run 10 trials of 10 episodes on each of the 5 held-out tasks. Meta-World resets are deterministic, so repeated trials would be identical, and those families run one trial per task. Every trial starts from an empty context. All methods act with the mean of their action head. The harness writes one record per (task, trial, episode) and averages nothing at write time.

The metrics are those of Section 3. For MetaWorld the harness logs, for every episode, the minimum and the final value of MetaWorld's object-to-goal distance (end effector to goal on Reach) and whether MetaWorld's success flag was raised at any step. Success@0.10 is the fraction of episodes 6 to 10 whose minimum distance is at most 0.10 m. Success@flag is the fraction whose success flag was raised; the flag requires a distance of at most 0.05 m on Reach and Push and 0.07 m on Pick-Place and Soccer. Because MetaWorld runs one trial per task, each success entry for one seed averages 25 binary outcomes (5 tasks and episodes 6 to 10), which are correlated within a trial. The spread over seeds is correspondingly wide. We report means and standard deviations over the six training seeds. Paired comparisons between rules are made per seed (Section 7).

B.5 COMPUTE

On one RTX 3090, a T2MIR-AD or AD run takes a median of 7.9 h and a T2MIR-DPT or DPT run 1.6 to 2.0 h. Exploratory prompting needs no training: a MuJoCo cell of 500 episodes (5 tasks, 10 trials, 10 episodes) takes 10 to 60 minutes. A prompt-dropout fine-tune of 20k steps takes 0.4 to 0.9 h on one A100 (Appendix G). Training the seven methods on the ten families took about 7,600 job-hours on RTX 3090 and A100 GPUs. Because jobs were packed onto shared cards, the GPU-hours counted once per card lie between 2,590 and 7,600. The prompt-dropout fine-tunes and their step-matched controls took about 160 job-hours and the evaluations about 700.

C THE SELF-CONTEXT LANDSCAPE

This appendix reports how every method ends a cold-start trial when it collects its own context, the configuration that exploratory prompting later modifies. Table 6 gives the last-5 return and Table 7 gives the two MetaWorld success measures. The HalfCheetah-Vel stuck rates are in Table 1. Every cell averages six training seeds.

Table 6: **Self-context last-5 return on the held-out tasks.** Mean ± standard deviation over training seeds. Every method starts from an empty context and follows its context rule as adapted in Appendix B. The last column is Prompt-DT prompted with a 5-step segment of a top-decile offline episode of the held-out task, a reference that does not start cold. Six seeds per cell.

	AD lineage		DPT lineage					reference
Family	T2MIR-AD	AD	T2MIR-DPT	DPT	Prompt-DT	Meta-DT	S-ICQL	oracle demo
HalfCheetah-Vel	−109.7±63.3	−128.1±50.3	−61.3±14.0	−92.4±35.6	−144.6±11.8	−253.3±39.6	−44.3±14.7	−50.3±2.0
Hopper-Mass	477.1±176.0	505.3±71.6	595.2±7.5	601.4±14.5	557.0±41.2	141.6±32.9	386.6±148.2	517.8±77.5
Hopper-Friction	559.5±48.6	583.0±11.9	579.0±21.2	581.3±10.1	552.7±13.8	196.3±36.2	353.8±171.6	560.7±33.6
Walker-Mass	303.8±181.0	154.9±219.9	711.0±39.5	720.6±59.2	601.7±46.4	274.1±71.3	450.4±79.2	655.9±117.2
Walker-Friction	511.2±173.5	382.1±275.4	768.6±10.5	771.7±5.8	752.3±10.1	285.1±24.4	647.3±104.4	750.8±42.3
Humanoid-Dir	693.6±124.1	641.6±53.3	688.7±69.6	599.7±67.2	665.4±38.0	331.7±81.2	439.7±101.3	699.0±47.8
Reach	1027.6±45.6	975.9±66.7	1258.5±36.8	1238.7±53.4	991.4±147.4	856.4±264.9	1207.7±127.1	1606.1±146.5
Push	866.3±362.3	958.6±542.2	526.6±277.8	988.0±264.3	827.3±346.0	338.9±392.9	1171.5±432.7	844.5±160.5
Pick-Place	782.5±105.8	601.2±288.5	818.1±125.8	758.5±107.4	484.3±305.6	3.6±1.4	738.9±327.4	925.8±55.8
Soccer	631.0±204.8	701.4±168.9	1207.7±251.7	1228.0±187.5	928.5±105.9	159.2±137.6	511.1±199.0	822.1±157.2

Table 7: **Self-context success on the held-out MetaWorld tasks.** Each entry is success@0.10 / success@flag. The first is the mean over training seeds of the fraction of episodes 6 to 10 whose minimum object-to-goal distance (end effector to goal on Reach) is at most 0.10 m. The second is the mean over training seeds of the fraction of those episodes on which MetaWorld's own success flag is raised. One trial per task, so each seed contributes 25 binary outcomes per measure. Six seeds per cell, the same for both measures.

	AD lineage		DPT lineage					reference
Family	T2MIR-AD	AD	T2MIR-DPT	DPT	Prompt-DT	Meta-DT	S-ICQL	oracle demo
Reach	0.36 / 0.04	0.29 / 0.01	0.55 / 0.03	0.57 / 0.00	0.46 / 0.27	0.45 / 0.11	0.57 / 0.09	0.97 / 0.47
Push	0.80 / 0.61	0.77 / 0.47	0.89 / 0.49	0.93 / 0.63	0.77 / 0.38	0.34 / 0.19	0.77 / 0.43	0.97 / 0.53
Pick-Place	0.59 / 0.41	0.55 / 0.35	0.73 / 0.35	0.70 / 0.28	0.63 / 0.33	0.00 / 0.00	0.50 / 0.40	1.00 / 0.80
Soccer	0.73 / 0.66	0.81 / 0.71	0.87 / 0.84	0.99 / 0.98	0.84 / 0.80	0.13 / 0.12	0.59 / 0.44	0.67 / 0.63

Where the gap to a demonstration lies. On HalfCheetah-Vel only S-ICQL ends above the oracle demo's -50.3 ± 2.0, at -44.3 ± 14.7; the next is T2MIR-DPT at -61.3 ± 14.0. The AD lineage ends at -109.7 ± 63.3 and -128.1 ± 50.3, and Meta-DT at -253.3 ± 39.6, with every one of its held-out trials stuck (Table 1). The large seed spread of the AD lineage on this family reflects the stuck mode rather than uniform noise: its per-seed stuck rates range from 0.02 to 0.54. On the MetaWorld families the largest gap is on Reach, where the best self-context return is 1258.5 ± 36.8 against 1606.1 ± 146.5 and success@0.10 is at most 0.57 against 0.97. Success@flag on Reach, whose flag radius is 0.05 m, is at most 0.11 for every method except Prompt-DT (0.27), against 0.47 for the oracle demo. The in-context methods therefore come near the hidden goal but seldom within MetaWorld's radius. This is not for lack of goal information in their own context (Figure 11 in Appendix D). Pick-Place shows the same pattern at a smaller scale, with success@0.10 of 0.73 for the best self-context method against 1.00.

Where it does not. The oracle demo is not an upper bound. Both DPT-lineage methods end above it on Hopper-Mass, Hopper-Friction, Walker-Mass, Walker-Friction and Soccer. On Push, DPT

(988.0 ± 264.3), S-ICQL (1171.5 ± 432.7) and both AD-lineage methods (866.3 and 958.6) end above the oracle demo's 844.5 ± 160.5, and only T2MIR-DPT, Prompt-DT and Meta-DT end below it. On Soccer, DPT's success is 0.99 / 0.98 against the oracle demo's 0.67 / 0.63. A five-step segment of a good offline episode is a strong prompt for some tasks and a weak one for others. We therefore use it as a reference point and measure exploratory prompting mainly against each method's own self-context run, paired by seed.

Differences between methods. A DPT-lineage method has the highest self-context last-5 return on six of the ten families. The exceptions are HalfCheetah-Vel and Push (S-ICQL), Hopper-Friction (AD) and Humanoid-Dir (T2MIR-AD). Meta-DT has the lowest last-5 return on nine families, all but Walker-Mass, including 3.6 ± 1.4 on Pick-Place with zero success.

Prompt-DT ends at one speed; S-ICQL recovers from a poor start. No stuck HalfCheetah-Vel trial of Prompt-DT returns the standing value in every episode. Its returns fit a policy that runs at one speed whatever the target. Write the mean per-step error on a held-out target as $e(\theta) = -G_{\text{last-5}}/200$. The fit $e(\theta) \approx |v^\star - \theta| + c$ gives $v^\star = 1.57$ m/s for Prompt-DT, with a root-mean-square residual of 0.05 m/s, against 0.48 for a constant error. Prompt-DT is accordingly stuck on 56 and 60 of the 60 trials at $\theta = 2.5$ and 3.0, and on none at 1.5. Because its stuck trials concentrate on a few targets, its per-seed stuck rate varies little (Table 1). S-ICQL, which re-encodes its previous episode, improves from a mean first-episode return of −238.9 to a last-5 return of −44.3 and ends stuck in 15 of its 300 trials.

A persistent low mode outside HalfCheetah-Vel. The AD lineage also has trials that stay low on Walker-Mass and Walker-Friction. These trials account for the lineage's large seed spreads on Walker-Mass, 303.8 ± 181.0 and 154.9 ± 219.9. We count a trial or an episode as low when its return is below half of the oracle-demo last-5 return. By that count, 398 of the 600 AD-lineage Walker-Mass trials end low. A low episode is followed by another low one with probability 0.95, against 0.02 after an episode that is not low. The low episodes end in a fall after a median of 30 steps, against 200 otherwise. On Walker-Friction the numbers are 234 of 600, 0.86 and 60 steps. EP lifts 273 of the 398 and 184 of the 234 low trials above the threshold and makes 29 of the other 202 and 41 of the other 366 low; these two families carry the three largest AD-lineage gains of Table 8. On Reach, 26 of the 60 AD-lineage trials end low, and a low episode is followed by another with probability 0.94 against 0.01. EP lifts 13 of the 26 and makes 6 of the other 34 low. The DPT lineage has 20 and 0 such trials on the two Walker families. Among its other families outside HalfCheetah-Vel, only Push has a comparable share (26 of 60, persistence 0.77 against 0.19), and EP lifts 20 of those low trials and makes 5 of the other 34 low. We dissect the mechanism only on HalfCheetah-Vel (Section 4); on the Walker families a low episode ends in an early fall.

D Full exploratory-prompting results

Table 8 reproduces every cell behind Table 2. The self and EP columns are last-5 returns, mean ± std over the six training seeds, and the paired gain is the mean ± s.e. of the per-seed difference. The two verdict columns apply the one-standard-error criterion of Section 7 and a two-standard-error criterion. The AUC columns are the 10-episode means. The last two columns give the fraction of stuck trials (last-5 return below −100) for HalfCheetah-Vel and success@0.10 for the four Meta-World families. The DPT lineage uses `always3` and the AD lineage a random first episode. Over the 40 pairs the tally is 26 wins, 12 ties and 2 losses at one standard error and $16/22/2$ at two.

Uncertainty of the paired gains. Figure 8 shows the per-seed gains of every pair with a hierarchical-bootstrap interval that resamples seeds and then the task and trial units within each seed. The interval lies above zero for 17 pairs and below zero for one (Walker-Mass DPT). A seed-level paired t interval, which treats the tasks as fixed, excludes zero for 14 pairs. No pair survives a Holm correction over the 40 pairs: with six seeds the smallest attainable Wilcoxon signed-rank p is 0.031, above the Holm threshold of $0.05/40$ for the first step. The mean gain is positive in 34 of the 40 pairs, 17 of 20 in each lineage. If the pairs were independent, a two-sided sign test would give $p = 8 \times 10^{-6}$, but the pairs share families and checkpoints. The one-standard-error verdicts of Table 2 therefore summarize direction. They are not significance claims.

Reading the result by family. Within the AD lineage, T2MIR-AD and AD gain +225.0 ± 59.3 and +370.4 ± 77.7 on Walker-Mass, where the self context often stays low (Appendix C). On Soccer,

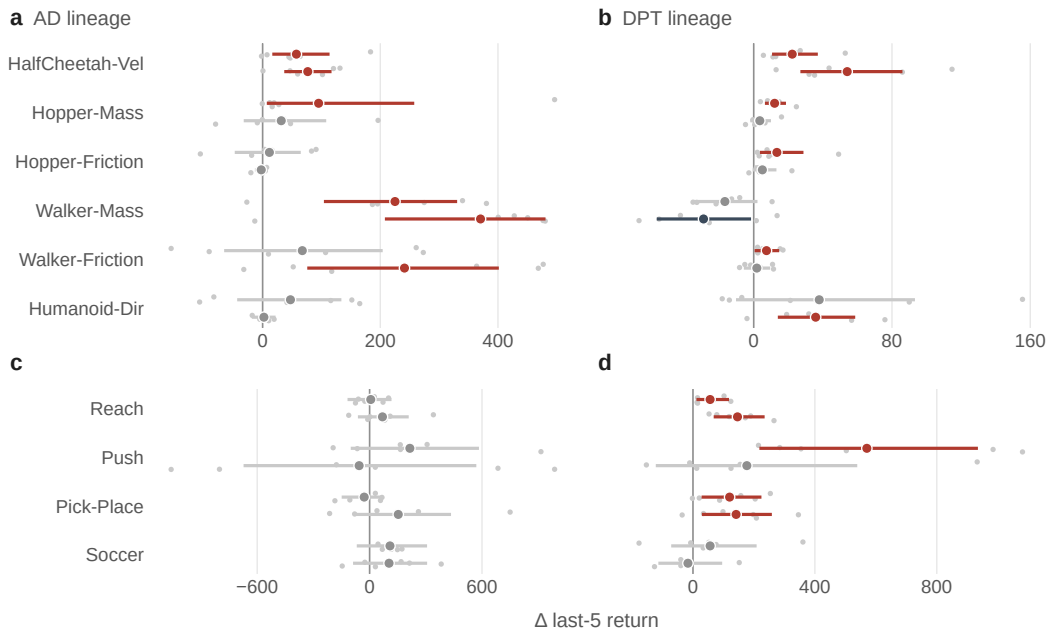


Figure 8: **EP minus self, paired last-5 return, per (family, method).** MuJoCo families in the AD (a) and DPT (b) lineage, MetaWorld families in the AD (c) and DPT (d) lineage, with separate scales; two rows per family, the T2MIR method above. Light dots, seeds; dot, mean; bar, hierarchical-bootstrap 95% interval (seeds, then task and trial units within seed). Red: interval above 0 (17 pairs); slate: below 0 (1, Walker-Mass DPT); gray: covers 0 (22). The interval treats the five held-out tasks as random, so it is wide on MetaWorld, which has one trial per task. Soccer T2MIR-AD is positive on 6 of 6 seeds, with a seed-level t interval of $[+58, +159]$, yet its bootstrap interval is $[-69, +307]$. The tally is $26/12/2$ at one standard error and $16/22/2$ at two. A seed-level paired t interval excludes 0 in 14 of 40 pairs.

where the DPT lineage ties, the AD lineage wins ($+108.8 \pm 19.6$ and $+103.8 \pm 77.0$) from lower self-context returns (631.0 and 701.4). Humanoid-Dir gains in the DPT lineage, from 644.2 to 681.1 pooled over the two methods. The AUC cost of the exploratory episode is visible on every DPT-lineage MuJoCo family without a stuck mode, including Hopper-Friction (568.6 to 517.5) and Walker-Friction (766.2 to 692.2). On HalfCheetah-Vel, by contrast, the AD-lineage AUC rises from -156.5 to -91.0 (Table 9). The oracle-demo reference is a different model: Prompt-DT prompted with a 5-step segment of a top-decile offline episode of the held-out task.

Per-episode curves. Figure 9 extends Figure 5 to all ten families, adds selection only and the oracle-demo level, and marks the random episode (its mean return is given in the caption). Most of the AUC cost of EP comes from this episode. A kept MetaWorld prompt replays exactly: with the deterministic reset, each of the 108 kept MetaWorld units returns the same value in every episode from 5 to 10, and no MuJoCo unit does. Figure 10 shows the AD lineage. On HalfCheetah-Vel, the AD-lineage self context climbs slowly, from -285.4 in episode 1 to -104.1 in episode 10. After the random first episode (-445.6), by contrast, the return is -51.1 in episode 2 and stays there. EP ends above self on 11 of 12 checkpoints (paired last-5 gain $+67.1 \pm 16.5$). Per-step Gaussian noise in every episode ends between the two on HalfCheetah-Vel (-93.6) and below EP on all 12 checkpoints. On Hopper-Mass, Hopper-Friction, Walker-Mass, Walker-Friction, Humanoid-Dir and Reach, the noise matches or beats the random first episode, and on Humanoid-Dir and Reach its last-5 return is higher by 120 and 143. On Push, Pick-Place and Soccer the random episode is better. The AD-lineage models thus make good use of a sorted buffer of four noisy episodes on most families, although on HalfCheetah-Vel noise makes a third of the non-stuck trials stuck (Appendix E).

Reach: the self context carries the goal. Figure 11 examines where the Reach gain comes from. With an empty context and a deterministic reset, a DPT-lineage checkpoint plays the same first episode on every task (12 of 12 checkpoints). The rewards of that episode nevertheless pin down the goal. Under MetaWorld's reach reward, the set of goals that reproduce all 200 logged rewards has a median volume of 1.2 cm^3 for self first episodes. For uniformly random episodes, which stay

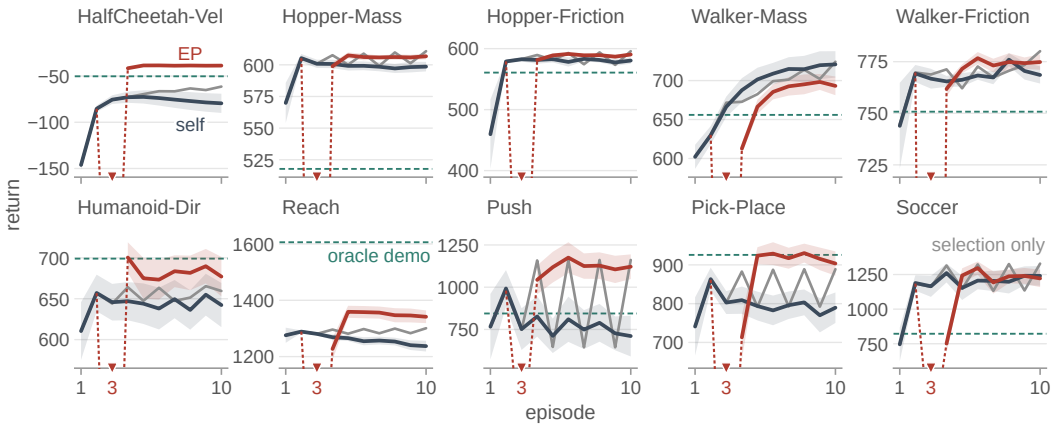


Figure 9: **Per-episode return in the DPT lineage on all ten families** (held-out tasks, mean over 12 checkpoints, bands ± 1 s.e., unpaired; Figure 5 shows four of these panels). EP (`always3`) makes episode 3 uniformly random (triangle on the bottom edge; mean return -445.0 on HalfCheetah-Vel, 0.5 to 355.7 elsewhere); most of the AUC cost of EP comes from this episode. Gray: selection only (`es1only`), which picks among the agent's own episodes by induced return. Dashed: the Prompt-DT oracle-demo level, a different model and a reference, not a ceiling.

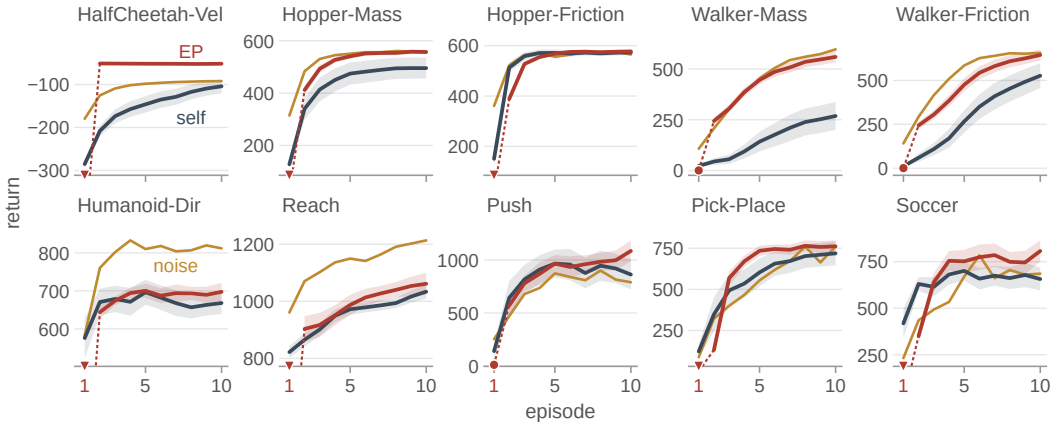


Figure 10: **Per-episode return in the AD lineage on all ten families** (T2MIR-AD and AD, six seeds each; mean over 12 checkpoints, bands ± 1 s.e., unpaired). EP is a uniformly random first episode: a triangle on the bottom edge where it lies below the axis, a dot at its value elsewhere (mean return -445.6 on HalfCheetah-Vel, 0.7 to 328.2 elsewhere). Ochre: Gaussian action noise ($\sigma = 0.3$) at every step of every episode. Noise matches or beats the random episode on Hopper-Mass, Hopper-Friction, Walker-Mass, Walker-Friction, Humanoid-Dir and Reach; the random episode is better on HalfCheetah-Vel (12 of 12 checkpoints), Push, Pick-Place and Soccer.

near the start where the reward is flat, the median volume is 35 cm^3. The random volume is the larger in 95% of (random, self) pairs (a). The frozen model uses this information only in part. Under the self context, final hand y follows goal y ($r = 0.69$), but final hand x does not follow goal x ($r = -0.18$). Under EP, final hand x follows goal x partly ($r = 0.41$, and 13 of 60 trials end more than 3 cm from the initial hand x on the goal's side). At the same time, the height error grows and the median final distance is unchanged (0.113 against 0.115 m; b). The gain in success within 0.10 m (DPT lineage, 0.56 to 0.72) is therefore a gain in the closest approach during an episode, not at its end. At MetaWorld's radius of 0.05 m, EP raises success from 0.013 to 0.167 in the DPT lineage (T2MIR-DPT 0.027 to 0.100, DPT 0.000 to 0.233; c). In the AD lineage, EP raises success at that radius from 0.027 to 0.030, raises success within 0.10 m (0.33 to 0.43) and lowers success within 0.15 m (0.92 to 0.85).

Paired runs. The self and EP arms are separate runs of the harness with the same checkpoints, tasks and resets. In the DPT lineage, episodes 1 and 2 follow the same rule in both arms. No unit (task and trial) differs there by more than 50 in return on HalfCheetah-Vel, the two Hopper families and

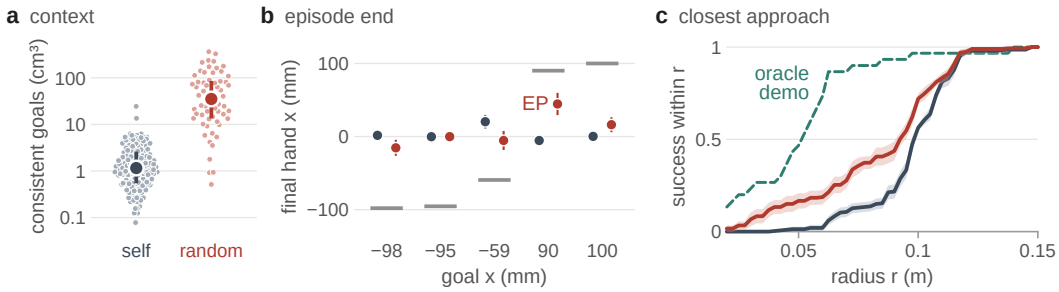


Figure 11: **Reach: the self context carries the goal, and EP helps within an episode rather than at its end** (DPT lineage in b and c). (a) Volume of the goals inside MetaWorld's goal space that reproduce all 200 rewards of one episode under the MetaWorld reach reward (tolerance: the misfit at the true goal plus 0.05), over 150 episode-goal pairs with a self first episode and 60 with a uniformly random episode, which stays near the start where the reward is flat; small dots, episode-goal pairs; large dot, median (1.2 and 35 cm^3); bar, interquartile range. The random volume is the larger in 95% of (random, self) pairs. (b) Final hand x at the end of episode 10 against goal x on the five held-out tasks (gray bar, the goal): per-task mean ± 1 s.e. over the 12 checkpoints, 60 trials per arm, under self (slate) and EP (red). Over trials, $r = -0.18$ under self and 0.41 under EP, while the median final distance is unchanged (0.115 m under self against 0.113 m under EP). (c) Success within r over episodes 6 to 10, mean over the 12 DPT-lineage checkpoints with a ± 1 s.e. band; dashed, Prompt-DT with the oracle demo (six checkpoints). EP raises success within 0.05 m from 0.013 to 0.167 and within 0.10 m from 0.56 to 0.72.

Reach. Chaotic episodes can diverge, however: 94 of 600 units differ by more than 50 on Walker-Mass, 84 on Walker-Friction and 9 on Humanoid-Dir, and 12, 2 and 1 of 60 on Push, Pick-Place and Soccer. The comparison is therefore paired by seed, task and reset, not by trajectory.

MetaWorld success flag. Success at MetaWorld's own threshold moves from self context to EP as follows, per method and family (T2MIR-AD, AD, T2MIR-DPT, DPT): Reach 0.04 → 0.03, 0.01 → 0.03, 0.03 → 0.10, 0.00 → 0.23; Push 0.61 → 0.59, 0.47 → 0.41, 0.49 → 0.59, 0.63 → 0.57; Pick-Place 0.41 → 0.35, 0.35 → 0.33, 0.35 → 0.47, 0.28 → 0.46; Soccer 0.66 → 0.71, 0.71 → 0.88, 0.84 → 0.89, 0.98 → 0.97. The self-context values are in Table 7.

Table 9 pools each lineage's two methods per family over their seeds and sets the result next to the Prompt-DT oracle-demo reference. This pooling is how Section 7.1 quotes lineage-level numbers.

Table 8: **Exploratory prompting, every (family, method) pair.** Last-5 return (mean ± std over six seeds), paired gain (mean ± s.e.), verdict at one and at two standard errors, 10-episode AUC, and a family-specific measure: the stuck rate on HalfCheetah-Vel and success@0.10 on MetaWorld. A · marks an entry that does not apply.

		last-5 return			verdict		AUC		stuck / s@0.10	
Family	Method	self	+EP	paired gain	1 s.e.	2 s.e.	self	+EP	self	+EP
HalfCheetah-Vel	T2MIR-AD	-109.7 ± 63.3	-52.3 ± 5.3	$+57.4 \pm 27.2$	win	win	-146.6	-91.4	0.257	0.043
	AD	-128.1 ± 50.3	-51.4 ± 7.4	$+76.7 \pm 20.4$	win	win	-166.4	-90.6	0.307	0.067
	T2MIR-DPT	-61.3 ± 14.0	-39.0 ± 3.0	$+22.3 \pm 7.0$	win	win	-71.9	-94.5	0.100	0.000
	DPT	-92.4 ± 35.6	-38.3 ± 2.1	$+54.1 \pm 15.6$	win	win	-95.5	-95.6	0.233	0.000
Hopper-Mass	T2MIR-AD	477.1 ± 176.0	572.2 ± 14.8	$+95.2 \pm 80.3$	win	tie	409.6	489.5	·	·
	AD	505.3 ± 71.6	537.0 ± 29.3	$+31.7 \pm 37.6$	tie	tie	442.9	463.1	·	·
	T2MIR-DPT	595.2 ± 7.5	607.3 ± 3.1	$+12.1 \pm 2.9$	win	win	596.4	545.9	·	·
	DPT	601.4 ± 14.5	605.0 ± 8.8	$+3.5 \pm 2.9$	win	tie	597.1	540.1	·	·
Hopper-Friction	T2MIR-AD	559.5 ± 48.6	571.0 ± 28.8	$+11.5 \pm 29.5$	tie	tie	510.0	491.7	·	·
	AD	583.0 ± 11.9	580.7 ± 7.2	-2.3 ± 3.9	tie	tie	533.9	495.7	·	·
	T2MIR-DPT	579.0 ± 21.2	592.4 ± 11.9	$+13.4 \pm 7.2$	win	tie	557.1	508.4	·	·
	DPT	581.3 ± 10.1	586.3 ± 4.6	$+5.0 \pm 3.5$	win	tie	580.1	526.7	·	·
Walker-Mass	T2MIR-AD	303.8 ± 181.0	528.8 ± 106.5	$+225.0 \pm 59.3$	win	win	204.9	395.7	·	·
	AD	154.9 ± 219.9	525.3 ± 58.7	$+370.4 \pm 77.7$	win	win	96.3	407.8	·	·
	T2MIR-DPT	711.0 ± 39.5	694.3 ± 28.4	-16.7 ± 7.0	loss	loss	686.2	604.1	·	·
	DPT	720.6 ± 59.2	691.6 ± 42.9	-29.0 ± 12.9	loss	loss	687.1	591.3	·	·
Walker-Friction	T2MIR-AD	511.2 ± 173.5	578.6 ± 107.4	$+67.5 \pm 72.9$	tie	tie	327.7	418.7	·	·
	AD	382.1 ± 275.4	623.2 ± 112.6	$+241.1 \pm 90.9$	win	win	242.3	464.4	·	·
	T2MIR-DPT	768.6 ± 10.5	776.0 ± 10.3	$+7.4 \pm 2.8$	win	win	765.9	692.8	·	·
	DPT	771.7 ± 5.8	773.5 ± 6.6	$+1.8 \pm 3.4$	tie	tie	766.5	691.5	·	·
Humanoid-Dir	T2MIR-AD	693.6 ± 124.1	740.9 ± 35.1	$+47.3 \pm 48.4$	tie	tie	685.0	657.6	·	·
	AD	641.6 ± 53.3	643.8 ± 50.1	$+2.2 \pm 5.3$	tie	tie	641.2	601.2	·	·
	T2MIR-DPT	688.7 ± 69.6	726.6 ± 36.1	$+37.9 \pm 28.7$	win	tie	687.9	658.6	·	·
	DPT	599.7 ± 67.2	635.5 ± 68.7	$+35.9 \pm 11.4$	win	win	597.1	574.5	·	·
Reach	T2MIR-AD	1027.6 ± 45.6	1033.8 ± 70.9	$+6.2 \pm 29.3$	tie	tie	973.8	910.2	0.36	0.46
	AD	975.9 ± 66.7	1044.5 ± 118.4	$+68.7 \pm 62.9$	win	tie	930.1	946.4	0.29	0.39
	T2MIR-DPT	1258.5 ± 36.8	1314.7 ± 15.6	$+56.2 \pm 20.1$	win	win	1268.6	1202.8	0.55	0.73
	DPT	1238.7 ± 53.4	1384.8 ± 92.7	$+146.1 \pm 32.1$	win	win	1256.0	1248.3	0.57	0.71
Push	T2MIR-AD	866.3 ± 362.3	1081.3 ± 185.2	$+215.0 \pm 158.1$	win	tie	759.1	877.2	0.80	0.87
	AD	958.6 ± 542.2	902.8 ± 342.3	-55.8 ± 327.8	tie	tie	848.6	754.5	0.77	0.83
	T2MIR-DPT	526.6 ± 277.8	1097.1 ± 209.9	$+570.5 \pm 151.8$	win	win	587.6	899.4	0.89	0.86
	DPT	988.0 ± 264.3	1164.8 ± 196.8	$+176.8 \pm 157.6$	win	tie	979.6	1015.2	0.93	0.88
Pick-Place	T2MIR-AD	782.5 ± 105.8	753.6 ± 117.5	-28.8 ± 41.0	tie	tie	668.6	598.5	0.59	0.57
	AD	601.2 ± 288.5	754.2 ± 94.1	$+153.0 \pm 137.8$	win	tie	443.6	577.0	0.55	0.57
	T2MIR-DPT	818.1 ± 125.8	938.3 ± 94.4	$+120.2 \pm 41.7$	win	win	814.3	795.5	0.73	0.84
	DPT	758.5 ± 107.4	900.1 ± 80.1	$+141.6 \pm 56.0$	win	win	775.9	772.4	0.70	0.89
Soccer	T2MIR-AD	631.0 ± 204.8	739.8 ± 227.9	$+108.8 \pm 19.6$	win	win	629.2	632.6	0.73	0.77
	AD	701.4 ± 168.9	805.2 ± 124.0	$+103.8 \pm 77.0$	win	tie	646.2	641.4	0.81	0.93
	T2MIR-DPT	1207.7 ± 251.7	1263.9 ± 164.5	$+56.2 \pm 71.2$	tie	tie	1150.5	1020.3	0.87	0.91
	DPT	1228.0 ± 187.5	1211.5 ± 197.6	-16.5 ± 37.3	tie	tie	1170.1	1004.9	0.99	0.97

Table 9: **Lineage means per family.** Last-5 return pooled over the six seeds of both methods of each lineage, with the 10-episode AUC in parentheses, under self context and under EP (`always3` for the DPT lineage, a random first episode for the AD lineage). The last column is Prompt-DT prompted with a 5-step segment of a top-decile offline episode of the held-out task (last-5).

	DPT lineage		AD lineage		
Family	self	+EP	self	+EP	oracle demo
HalfCheetah-Vel	-76.9 (-83.7)	-38.7 (-95.0)	-118.9 (-156.5)	-51.8 (-91.0)	-50.3
Hopper-Mass	598.3 (596.7)	606.1 (543.0)	491.2 (426.2)	554.6 (476.3)	517.8
Hopper-Friction	580.2 (568.6)	589.4 (517.5)	571.2 (522.0)	575.9 (493.7)	560.7
Walker-Mass	715.8 (686.6)	692.9 (597.7)	229.3 (150.6)	527.0 (401.7)	655.9
Walker-Friction	770.1 (766.2)	774.7 (692.2)	446.6 (285.0)	600.9 (441.6)	750.8
Humanoid-Dir	644.2 (642.5)	681.1 (616.6)	667.6 (663.1)	692.4 (629.4)	699.0
Reach	1248.6 (1262.3)	1349.8 (1225.5)	1001.7 (951.9)	1039.1 (928.3)	1606.1
Push	757.3 (783.6)	1130.9 (957.3)	912.5 (803.8)	992.0 (815.8)	844.5
Pick-Place	788.3 (795.1)	919.2 (783.9)	691.8 (556.1)	753.9 (587.8)	925.8
Soccer	1217.8 (1160.3)	1237.7 (1012.6)	666.2 (637.7)	772.5 (637.0)	822.1

E RULE DEFINITIONS AND ABLATIONS

Conventions. Every trial has $K = 10$ episodes of $H = 200$ steps and starts from an empty context. Unless a rule says otherwise, every action comes from the frozen deterministic head, $a_t = \mu_\phi(s_t, C_k)$. A *random episode* draws a_t i.i.d. from the uniform distribution on the normalized action box $\mathcal{A} = [-1, 1]^{d_a}$. The random episode's states, actions and rewards are recorded and enter the context exactly as any other episode does. The *induced return* of a prompt C is $G(\tau)$ for the episode τ acted under C. The released DPT-lineage rule is rolling, $C_{k+1} = \tau_k$ with $C_1 = \varnothing$. The prompt is frozen within an episode and is not sorted. The released AD-lineage rule keeps four episodes: when a fifth finishes, the lowest-return episode of the previous buffer is dropped and the four are re-sorted by return, ascending. *self* denotes the released rule with no change.

DPT-lineage rules.

- `always3` (Algorithm 1). Episodes 1 and 2 follow the rolling rule. Episode 3 is random. Episode 4 acts with $C_4 = \tau_3$. If $G(\tau_4) \geq G(\tau_2)$, the random episode is kept as the fixed prompt, $C_k = \tau_3$ for $k = 5, \ldots, 10$, and finished episodes are discarded. Otherwise $C_5 = \tau_2$ and the rolling rule resumes, $C_{k+1} = \tau_k$ for $k \geq 5$.
- `adaptive10`. As `always3`, except that episode 3 is random only if $G(\tau_2) \leq G(\tau_1) + 0.1\,|G(\tau_1)|$, that is, if episode 2 did not improve on episode 1 by more than 10%. Otherwise the rolling rule runs for the whole trial.
- `rand1`. Episode 1 is random. The rolling rule follows, so $C_2 = \tau_1$ and $C_{k+1} = \tau_k$.
- `rand1keep`. Episode 1 is random and stays the prompt for the whole trial, $C_k = \tau_1$ for $k \geq 2$.
- `es11only`, a $(1{+}1)$ prompt selection without exploration. After episode 1 the incumbent prompt is $P^\star = \tau_1$. Even episodes $k = 2, 4, \ldots, 10$ exploit: they act under $P^\star$, and their return becomes the incumbent's current value $v^\star = G(\tau_k)$. Odd episodes $k = 3, 5, 7, 9$ test: they act under the candidate τ_{k-1}, the exploit episode that has just finished. The candidate becomes the incumbent, with $v^\star = G(\tau_k)$, if and only if $G(\tau_k) > v^\star$. Otherwise $P^\star$ is kept.
- `rand1es11`. As `es11only`, with a random episode 1, which is therefore the first incumbent.
- `sig03`. In every episode, $a_t = \mathrm{clip}_{\mathcal{A}}\big(\mu_\phi(s_t, C_k) + 0.3\,\varepsilon_t\big)$ with $\varepsilon_t \sim \mathcal{N}(0, I)$ i.i.d. over steps, under the released context rule. This is a control, not a proposed rule.

AD-lineage rules.

- `rand1`, the AD-lineage rule of the main results. Episode 1 is random. The released sorted-buffer rule follows unchanged.
- `rand3`. Episodes 1 and 2 follow the released rule, episode 3 is random, and the released rule follows.
- `sig03`. Per-step Gaussian noise as in the DPT lineage.

The AD-lineage adapters have no single prompt to select. AD-lineage cells launched under the DPT-lineage selection rules with a random first episode therefore reduce to `rand1` and are pooled with it.

Design notes. When the exploratory prompt loses the comparison of `always3`, the trial continues the self chain as if episodes 3 and 4 had not happened, with τ_2 as the prompt of episode 5. With deterministic dynamics and reset, this branch reproduces the unmodified trial delayed by two episodes. With MuJoCo reset noise it does so only approximately. Exploring at episode 3 rather than 1 leaves episodes 1 and 2 exactly as the released rule produces them and supplies the reference value $G(\tau_2)$ of the self chain. Per-step noise is paid in every episode, including those that should exploit, while the random episode is paid once. In the AD lineage the random episode leaves the buffer at the first overflow whenever it has the lowest return of the four, which is the case in nearly every trial.

Paired comparison. For each rule and each (family, method) pair we take the training seeds evaluated under both the rule and the released rule: six for every pair. For each seed we form the

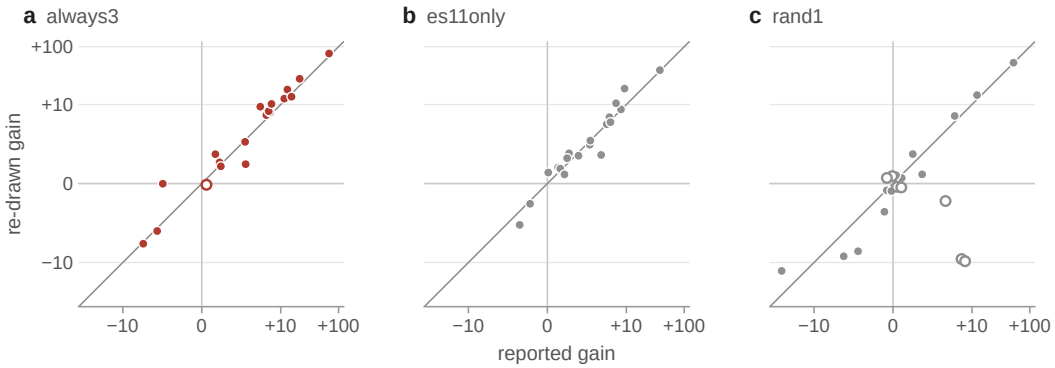


Figure 12: **The rule gains under a second evaluation draw.** Normalized last-5 gain over self per (family, method) pair, DPT lineage, six training seeds: the reported evaluation (x) against a re-evaluation of the same checkpoints and held-out tasks with new resets and new random episodes (y), for `always3` (a), `es11only` (b) and `rand1` (c). Filled, the sign agrees under both draws; hollow, it changed. Diagonal, equal gain. Symmetric log axes, linear within ± 2. The sign agrees in 19, 20 and 13 of the 20 pairs; the means over the pairs are given in the paired-comparison paragraph of Appendix E.

difference in last-5 return, with each term averaged over held-out tasks and trials. The standard error of the mean is the sample standard deviation over seeds divided by the square root of their number. The pair is a win if the mean difference exceeds one standard error, a loss if it is below minus one standard error, and a tie otherwise. Table 10 averages the mean paired difference over the 20 pairs of each lineage, for last-5 return and for the 10-episode AUC. Each average is reported in return units and in units normalized per family as in Figure 6 (100 = the gap from a uniformly random episode to the oracle demo). The table's last two columns count, on HalfCheetah-Vel, the self-stuck trials that the rule releases and the trials that are not stuck under self but become stuck under the rule. The rules were designed and compared on the held-out tasks on which they are reported, so the ranking among them is subject to selection. A re-evaluation of all six DPT-lineage seeds with new resets and new random episodes (base seed 20270905) keeps the sign of the normalized last-5 gain in 52 of the 60 (rule, pair) combinations: 19 of 20 for `always3`, 20 of 20 for `es11only` and 13 of 20 for `rand1` (Figure 12). The mean over the 20 pairs is $+8.6$ reported against $+10.2$ re-drawn for `always3`, $+4.5$ against $+5.1$ for `es11only`, and $+2.4$ against $+1.4$ for `rand1`. Every sign change on MuJoCo lies within 0.4 normalized units of zero under both draws. The larger sign changes are on Reach, Push and Soccer, where each held-out task has one trial. The re-evaluation tests luck in the evaluation draw, not the choice of rule.

Reading the DPT-lineage rows. `always3` has the most wins, shared with `es11only`, and the largest mean last-5 gain, $+69.9$. Its mean AUC change is negative, -32.7, because episode 3 is spent on random actions and episode 4 on testing the resulting prompt (Section 7.2). `es11only` is the only DPT-lineage rule with a positive mean AUC change, $+22.5$, and it wins as often as `always3`. Its mean last-5 gain, however, is about half as large, $+36.8$. Its candidates are the model's own episodes, which in a stuck DPT-lineage trial all overshoot. Keeping an earlier, slower one can stop the chain from accelerating, and the rule releases 26 of the 100 stuck HalfCheetah-Vel trials. Adding a random first episode to the same selection (`rand1es11`) releases all 100 yet lowers the number of wins to 12 and the mean AUC change to -33.9. That is the cost on tasks with an improving self chain, and it is why we explore at episode 3. `adaptive10` explores only when episode 2 did not improve by more than 10%, so it misses chains that improve briefly and then fall. It releases 55 of the 100 and wins 12 pairs. The two rules without selection are the weakest of the exploratory rules. Under `rand1` the rolling rule replaces the random prompt after one episode. The rule releases 95 of the 100 stuck HalfCheetah-Vel trials, but most pairs are ties ($4/15/1$). Keeping the random episode as the prompt for the whole trial (`rand1keep`) gives up whatever the model's own later episodes would add and loses six pairs. Per-step noise (`sig03`) loses 11 of 20 pairs, releases 53 of the 100 stuck HalfCheetah-Vel trials and makes 137 of the other 500 stuck. Per family (Table 11), `always3` has a higher mean last-5 return than `es11only` on eight families. The 95% interval of the paired difference over seeds lies above zero only on HalfCheetah-Vel, Reach and, marginally,

Table 10: **Every training-free context rule against the released rule.** Paired over training seeds within each (family, method) pair, 20 pairs per lineage. A win or loss is a mean paired difference more than one standard error above or below zero. Δ: mean over pairs of the paired change, in return units and normalized per family as in Figure 6. Last two columns, HalfCheetah-Vel: self-stuck trials released by the rule, and trials not stuck under self that the rule makes stuck (of 100 and 500 in the DPT lineage, 169 and 431 in the AD lineage). Twelve `adaptive10` cells were rerun after a fix to the rule's implementation.

				Δ return		Δ normalized		HalfCheetah-Vel	
Lineage	Rule	What the agent does	W / T / L	last-5	AUC	last-5	AUC	released	made stuck
DPT	`always3`	Random episode 3, kept as the prompt if it induces at least episode 2's return	15 / 3 / 2	+69.9	−32.7	+8.6	−5.2	100	0
DPT	`rand1es11`	Random episode 1, then (1+1) selection by induced return	12 / 7 / 1	+45.8	−33.9	+5.8	−5.0	100	1
DPT	`es11only`	(1+1) selection by induced return, no random episode	15 / 4 / 1	+36.8	+22.5	+4.5	+2.7	26	4
DPT	`adaptive10`	Random episode 3 only if episode 2 gained at most 10% over episode 1	12 / 6 / 2	+26.9	−27.7	+3.2	−4.0	55	2
DPT	`rand1`	Random episode 1, then the rolling self prompt	4 / 15 / 1	+18.0	−53.0	+2.4	−7.3	95	4
DPT	`rand1keep`	Random episode 1 kept as the prompt for the whole trial	7 / 7 / 6	−10.7	−74.9	−1.1	−10.0	100	4
DPT	`sig03`	Gaussian action noise, $\sigma = 0.3$, at every step	3 / 6 / 11	−33.6	−39.0	−5.3	−6.1	53	137
AD	`rand1`	Random episode 1 enters the sorted buffer (EP rule)	11 / 9 / 0	+89.7	+48.1	+13.3	+7.8	152	16
AD	`rand3`	Random episode 3 enters the sorted buffer	7 / 10 / 3	+30.8	−25.7	+5.6	−2.8	147	5
AD	`sig03`	Gaussian action noise, $\sigma = 0.3$, at every step	11 / 7 / 2	+90.0	+80.9	+13.0	+12.3	132	143

Humanoid-Dir, and below zero on Walker-Mass (Figure 6c). `es11only` has the higher AUC on nine families, all but Push.

Reading the AD-lineage rows. In the AD lineage, per-step noise matches a random first episode on last-5 return ($11/7/2$ against $11/9/0$; $+13.0$ against $+13.3$ normalized) and is higher on AUC ($+12.3$ against $+7.8$). The reason is that noise costs no episode and that the model still makes good use of a sorted buffer of four noisy episodes on the dynamics families (Figure 10). Noise is not a safe substitute on HalfCheetah-Vel: it releases fewer stuck trials than the random episode (132 against 152 of 169) and makes 143 of the 431 non-stuck trials stuck, against 16. Its stuck trials are not standing ones: noise ends no trial standing in either lineage, and at -100 these trials are stuck because noise lowers the return of every episode. At a threshold of -150, noise leaves 2.7% to 7.3% of trials stuck in the four methods, against 0% to 1.3% for EP. Of self, noise and EP, EP has the lowest stuck share at each of the thresholds -75, -100 and -150. The two lineages therefore agree about one whole exploratory episode and disagree about per-step noise. On the DPT-lineage probe checkpoint of Appendix F.1, a constant per-episode action offset leaves more trials stuck than the deterministic head does. There too, what helps is an exploratory episode in the context rather than perturbed actions. Exploring at episode 3 instead of 1 (`rand3`) releases about as many stuck HalfCheetah-Vel trials (147 of 169) and makes fewer others stuck (5). It gains less elsewhere, however: 7 wins, 10 ties and 3 losses, a mean last-5 gain of $+30.8$ against $+89.7$, and a lower mean AUC change (-25.7 against $+48.1$).

Which exploratory episode, and whether the gains hold on training tasks. In Figure 13, other whole episodes replace the uniformly random episode of `always3` under the same selection, on all twelve DPT-lineage checkpoints. A Gaussian ($\sigma = 1$) or ε-greedy ($\varepsilon = 0.5$) episode releases most stuck HalfCheetah-Vel trials, though fewer than EP (3.3% and 4.2% of trials stuck, against 0% for EP and 16.7% for self). On HalfCheetah-Vel, such an episode beats EP in last-5 return on at most one of the twelve checkpoints, but it costs less AUC. An early offline episode of another training task helps less (12.0% stuck). On the checkpoints tested, what matters is one whole exploratory episode followed by selection, not uniform actions in particular, though the uniform episode releases the most stuck trials. A same-task demonstration or SAC-expert context does better still on HalfCheetah-Vel and Reach, but it is not available at a cold start. Panels b and c ask whether the failure is specific to held-out tasks. Per checkpoint, the last-5 returns on training and held-out tasks are correlated ($r = 0.90$ under self, 0.78 under EP). The mean normalized EP gain of each (family, lineage) has the same sign on training as on held-out tasks in 19 of 20 cells (Pearson $r = 0.96$ across cells); the exception, Hopper-Friction in the AD lineage, lies within 0.02 of zero on both. On

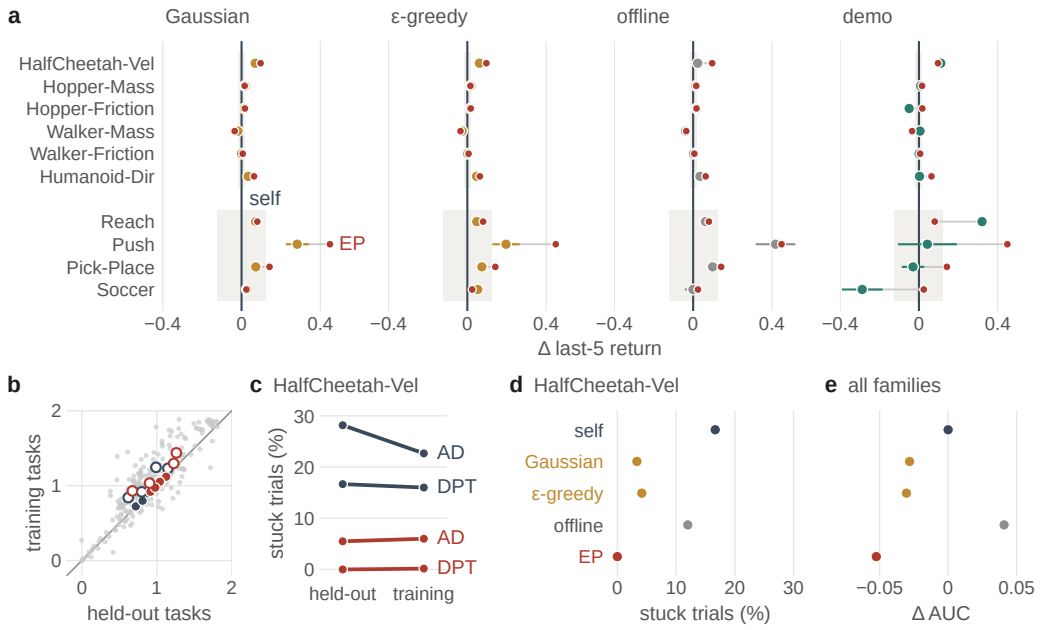


Figure 13: **Which exploratory episode, and whether the gains hold on training tasks.** (a) Mean change in normalized last-5 return relative to self (vertical line; fraction of the random-to-demo gap) over the twelve DPT-lineage checkpoints of each family on held-out tasks: EP (red) against the same checkpoints given a whole Gaussian ($\sigma = 1$) or ε-greedy ($\varepsilon = 0.5$) exploratory episode under the same selection, an early offline episode of another training task, or a same-task demonstration (teal); whiskers, $\pm$ s.e. over checkpoints; band, rerun spread of the same checkpoint and rule (MuJoCo ± 0.018, MetaWorld ± 0.123). EP closes 87% of the self-to-demo gap on HalfCheetah-Vel and 25% on Reach, and ends above the demo on Push. Not drawn: a SAC-expert context on the six T2MIR-AD checkpoints ends at -36.2 on HalfCheetah-Vel and 1373.9 on Reach, against -52.3 and 1033.8 for EP on the same checkpoints. (b) Normalized last-5 on training against held-out tasks: gray, checkpoints; large dots, family means under self (slate) and EP (red), hollow for MetaWorld; within-family $r = 0.90$ (self) and 0.78 (EP) over checkpoints. Training tasks score higher on MetaWorld ($+0.170$ under self) but not on MuJoCo ($+0.006$). (c) HalfCheetah-Vel stuck share on held-out and training tasks, twelve checkpoints per lineage: self 28.2% and 22.7% (AD), 16.7% and 16.0% (DPT); EP 5.5% and 6.0% (AD), 0% and 0.2% (DPT). (d) HalfCheetah-Vel stuck share of the same twelve DPT-lineage checkpoints under each exploratory episode: Gaussian 3.3%, ε-greedy 4.2%, other-task offline 12.0%, self 16.7%, EP 0%; a same-task demonstration also leaves 0% stuck. (e) Change in normalized 10-episode AUC relative to self over the 120 checkpoints of all families: the noisy episodes cost less than EP (-0.028 and -0.030 against -0.052), and the other-task offline episode raises AUC ($+0.041$).

MuJoCo the training tasks score no higher than the held-out ones ($+0.006$ of the random-to-demo gap). On HalfCheetah-Vel the self-context stuck share is 22.7% on training tasks against 28.2% on held-out ones in the AD lineage, and 16.0% against 16.7% in the DPT lineage (twelve checkpoints each). There the attractor is therefore not a failure to generalize. EP cuts the training-task shares to 6.0% and 0.2%. On MetaWorld the training tasks score higher ($+0.170$), so the MetaWorld results mix the cold start with generalization to held-out goals.

Table 11: **Per family, exploring at episode 3 against selection without exploration (DPT lineage).** Last-5 return (AUC over the ten episodes in parentheses), mean over both methods and six seeds each. Bold, the higher last-5 mean. `always3` has the higher mean on eight families. The paired advantage is clear only on HalfCheetah-Vel, Humanoid-Dir and Reach, and it reverses on Walker-Mass (Figure 6c). `es11only` has the higher AUC on nine (all but Push).

Family	`always3`	`es11only`
HalfCheetah-Vel	**−38.7** (−95.0)	−64.6 (−77.3)
Hopper-Mass	**606.1** (543.0)	606.0 (601.3)
Hopper-Friction	**589.4** (517.5)	586.1 (572.3)
Walker-Mass	692.9 (597.7)	**708.1** (680.0)
Walker-Friction	**774.7** (692.2)	772.8 (768.1)
Humanoid-Dir	**681.1** (616.6)	657.4 (651.0)
Reach	**1349.8** (1225.5)	1292.8 (1288.7)
Push	**1130.9** (957.3)	954.4 (906.9)
Pick-Place	**919.2** (783.9)	850.0 (833.0)
Soccer	1237.7 (1012.6)	**1248.9** (1179.8)

F Diagnosis: probes, speed logs, context swaps and distance to the pretraining data

This appendix gives the evidence behind Section 4 in detail. The first subsection reports probes on single checkpoints. The next three report the trajectory logs behind Figure 2, every context swap behind Figure 3, and where first episodes lie relative to the pretraining data. The last two test, on one T2MIR-AD checkpoint, two alternatives to changing the context: activation steering, which shows that the frozen model encodes the HalfCheetah-Vel target speed linearly, and policy-gradient post-training on the model's own trials, which did not remove the stuck mode. Throughout, a trial is stuck if its last-5 return is below -100.

F.1 Probe results

The probes use single HalfCheetah-Vel checkpoints and the environment seeds of Appendix B. We take the most stuck seed of T2MIR-DPT (seed 4, stuck rate 0.24) and the most stuck seed of T2MIR-AD (seed 0, 0.52; Table 1).

Returns alone do not identify the attractor. Over all seeds and held-out HalfCheetah-Vel trials under the self context, the ten returns of a stuck trial span at most 10% of their mean magnitude in 94 of the 169 stuck AD-lineage trials and in 3 of the 100 stuck DPT-lineage ones. Flat returns do not mean standing still. Running at 2θ earns the same return as standing. Two of the three flat DPT-lineage trials return within 15% of -200θ in every episode, yet no DPT-lineage stuck trial starts standing. Section 4 therefore classifies episodes from logged speeds (Appendix F.2).

Table 12: **Diagnosis probes on HalfCheetah-Vel.** Held-out tasks, 5 tasks × 5 trials × 10 episodes per row. Returns are means over the 25 trials. The curve lists the mean return of episodes 1 to 10. Per-step noise is Gaussian with $\sigma = 0.3$, clipped to the action bounds, in every episode. The per-episode offset is a constant action offset drawn once per episode with $\sigma = 0.3$, plus white noise with $\sigma = 0.05$. The $(1 + 1)$ selection rule is defined in Appendix E. The expert context holds four SAC-expert episodes of the task.

Context or perturbation	episode 1	last-5	stuck	mean return, episodes 1 to 10
T2MIR-DPT, seed 4				
self context, deterministic	−144.9	−96.9	7/25	−145 −83 −77 −83 −86 −92 −95 −98 −99 −100
per-step noise, every episode	−152.7	−87.7	8/25	−153 −93 −89 −83 −81 −84 −87 −92 −87 −89
per-episode action offset	−208.5	−109.4	14/25	−209 −119 −101 −108 −116 −111 −105 −114 −101 −116
random episode 1, then self context	−446.9	−40.0	0/25	−447 −39 −38 −38 −38 −38 −39 −40 −41 −42
random episode 1 kept as the prompt	−446.9	−40.9	0/25	−447 −39 −40 −41 −40 −41 −40 −42 −41 −40
random episode 1, then $(1 + 1)$ selection	−446.9	−35.5	0/25	−447 −39 −38 −38 −37 −36 −36 −36 −35 −35
T2MIR-AD, seed 0				
self context, deterministic	−249.4	−236.0	12/25	−249 −250 −251 −250 −250 −235 −236 −237 −236 −236
expert context (ceiling)	−32.7	−32.9	0/25	−33 −33 −33 −34 −32 −33 −33 −33 −32 −33

Three observations follow from Table 12. First, the AD-lineage failure is flat from the first episode: the mean return of the self context stays between -251 and -235 over the ten episodes, while the expert context holds the same checkpoint at about -33 throughout. Second, the DPT-lineage self

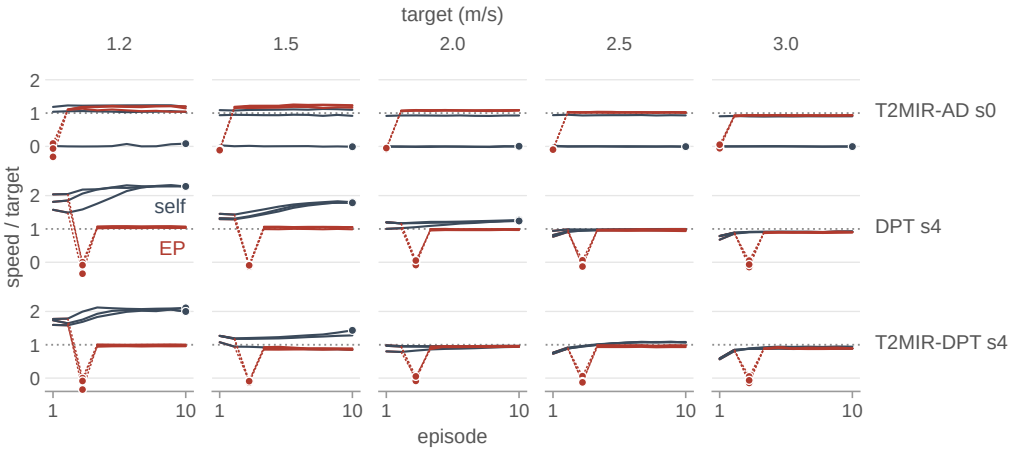


Figure 14: **All logged HalfCheetah-Vel trials of three checkpoints** (rows: T2MIR-AD seed 0 and DPT seed 4, shown in Figure 2a, and T2MIR-DPT seed 4; columns: the five targets, three trials each). Speed over target per episode under self (slate) and EP (red); red dot, the random episode; dotted, the target; slate end dot, a self trial that ends stuck (8, 9 and 4 of 15; none under EP). The self runs equal the grid runs. For targets above 1.2 m/s the EP runs draw different random episodes from the grid's: one uniform-action stream runs through all trials, and the logged runs play three trials per target where the grid plays ten.

curve improves for two episodes and then declines, the overshooting chain of Section 4. Third, every row that places a uniformly random episode in the context removes all stuck trials, whether the episode is then replaced by the model's own episodes, kept as the prompt, or compared with them by induced return. Neither form of action noise removes them.

Attractor probe on training tasks. A second probe follows individual trials of T2MIR-AD seed 0 on two training tasks with target speeds 2.10 and 2.90 m/s, three trials of five episodes each. Under the self context, the stuck trials return -429 to -422 (target 2.10) and -585 to -579 (target 2.90) in every episode, with mean forward speed between -0.02 and 0.03 m/s. The return of standing still is -200 times the target speed, as in (2). The non-stuck trials run at 2.07 to 2.10 and 2.75 to 2.79 m/s from episode 1. Inserting one SAC-expert episode into the context after episode 1 rescues all three stuck trials at episode 2 ($-425 \to -24$, $-422 \to -23$, $-579 \to -43$). A uniformly random first episode puts 6 of 6 trials at the target from episode 2 (for example $-477 \to -76$ and $-649 \to -43$, where the first number is the random episode). Gaussian noise with $\sigma = 0.3$ in episode 1 alone leaves 5 of the 6 trials at the target, rescuing two of the three stuck ones. The remaining trial, at target 2.90, stays at -595 to -586. These stuck trials stand still, and they do so on training tasks, so on this checkpoint the failure is not one of generalizing to held-out targets (Figure 13c).

F.2 Speed logs

We reran all 24 HalfCheetah-Vel checkpoints (six seeds of each of the four methods) with the grid's resets and per-step logging of speeds and actions. The reruns reproduce the grid: on 20 checkpoints all 50 trials keep their stuck label, and on the other four 48 or 49 do. Section 4 and Figure 2b use the stuck trials whose rerun reproduces the grid's episode-1 return to within 1 (168 of 169 AD-lineage, 88 of 100 DPT-lineage; the reruns of DPT seed 2 and T2MIR-DPT seed 4 diverge most). The counts below and in Figure 17 use the reruns' own labels (169 and 102). We also logged three checkpoints for the traces of Figure 2a. Figure 14 shows all 15 logged trials of each under self and EP, so that the choice of the trial in Figure 2a can be checked. Figure 15 plots every consecutive pair of self-context returns in the grid. In the AD lineage, 1,176 of the 1,521 return pairs of stuck trials lie within 5% of -200θ, the return of standing still. In the DPT lineage 27 of 900 do, all at $\theta = 1.2$ or 1.5. In that lineage 76% of the stuck trials' pairs lie below the diagonal, each episode worse than the one before, against 35% for the trials that are not stuck. Under EP the episode after the random one is above -100 in 159 of the 169 self-stuck AD-lineage trials and in 100 of 100 DPT-lineage ones.

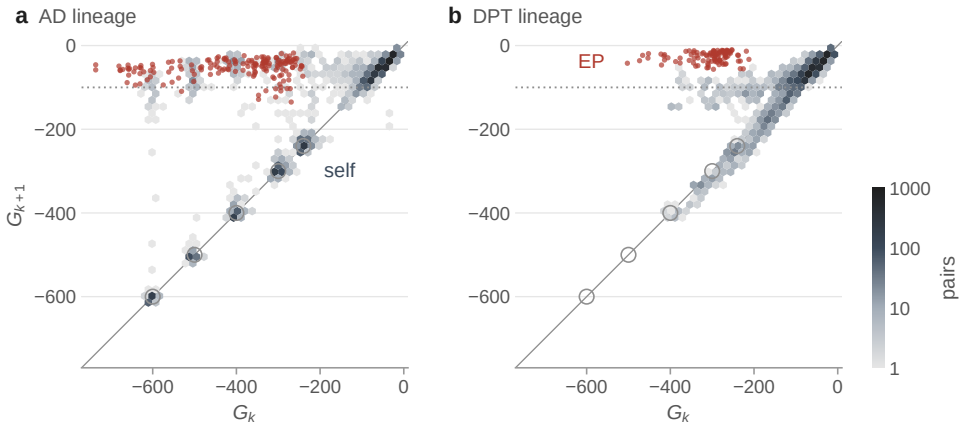


Figure 15: **Every consecutive pair of self-context returns** (G_k, G_{k+1}) in the HalfCheetah-Vel grid, AD (a) and DPT (b) lineage (600 trials and 5,400 pairs each), as a count per hexagon on a log scale. Circles, -200θ for the five held-out targets, the return of standing still and also of running at 2θ; such pairs are rare in the DPT lineage (27 of 5,400), where no logged trial starts standing (Figure 17). Dotted, the stuck threshold -100. Red: for each self-stuck trial, the random episode and the next episode under EP. The next episode clears the threshold in 159 of 169 AD-lineage and 100 of 100 DPT-lineage trials; by last-5 return EP releases 152 of 169 and 100 of 100 and makes 16 of 431 non-stuck AD-lineage trials stuck.

F.3 Context swaps

Design. We use T2MIR-AD seeds 0, 3 and 4 and T2MIR-DPT seed 4, chosen before the experiment for their high self-context stuck rates. A second batch adds AD seed 1 and DPT seeds 0 and 4, among the most stuck seeds of their methods (stuck rates 0.38, 0.30 and 0.56). Each arm replays the same 25 resets (5 held-out tasks $\times$ 5 trials) and replaces only the first context episode. Everything after it follows the released rule. The 68 trials stuck under the self context fall in 24 (checkpoint, task) cells. Trials of a deterministic context within a cell are near replicates, so the effective sample is smaller than the trial count. The context episodes are: a uniformly random episode; *still*, an executed episode of zero actions; *expert*, a SAC-expert episode of the held-out task; *early*, an early episode of the held-out task's SAC run (mean return -280); and *tier* q, the episode at chronological quantile q of that run ($q = 0$: its first checkpoint). We planned a standing self episode from another trial as an arm and ran the zero-action episode in its place. The expert, early and tier episodes come from the held-out task's own data, not from pretraining, and carry rewards a deployed agent would not have. They probe what the model reads and are not a usable remedy. In the θ' arms the rewards of the random or early episode are relabeled for a wrong target ($\theta' = 2.9$ for $\theta \leq 2$, 1.1 otherwise). No zero-action arm is relabeled.

Results (Figure 16). Under a θ' arm none of the 68 self-stuck trials stands: they run at the wrong speed, and 62 (random) and 56 (early) of them end stuck. Every tier $q \geq 0.1$ leaves 0 of 25 trials stuck on every checkpoint, and the induced last-5 return is flat in q, between -42.3 and -41.1 on average from $q = 0.1$ to 1. The return of the context episode itself rises from -74 to -28. The $q = 0$ episode (return -416) leaves 6, 4 and 2 of 25 trials stuck on T2MIR-AD seeds 0, 3 and 4, 12 on AD seed 1 and 0 on the three DPT-lineage checkpoints. A random episode releases 59 of the 68, and the most it leaves is 5 of 25, on DPT seed 4 (15 under self). Neither lineage trains on checkpoint order (Appendix B), so q indexes the data, not a training order. In 105 of the 175 trials of panels c and d the early context episode stands, and every early trial that ends stuck had a standing context. With the early episode relabeled for θ', 149 of 175 trials move toward θ' and 113 end closer to θ' than to θ.

F.4 Distance to the pretraining data

Figure 17 asks where first episodes lie relative to the 7,410 pretraining episodes of HalfCheetah-Vel, on the 24 logged checkpoints. Per episode (a), the stuck AD-lineage trials that start standing vary their actions less than almost every pretraining episode (median within-episode action standard

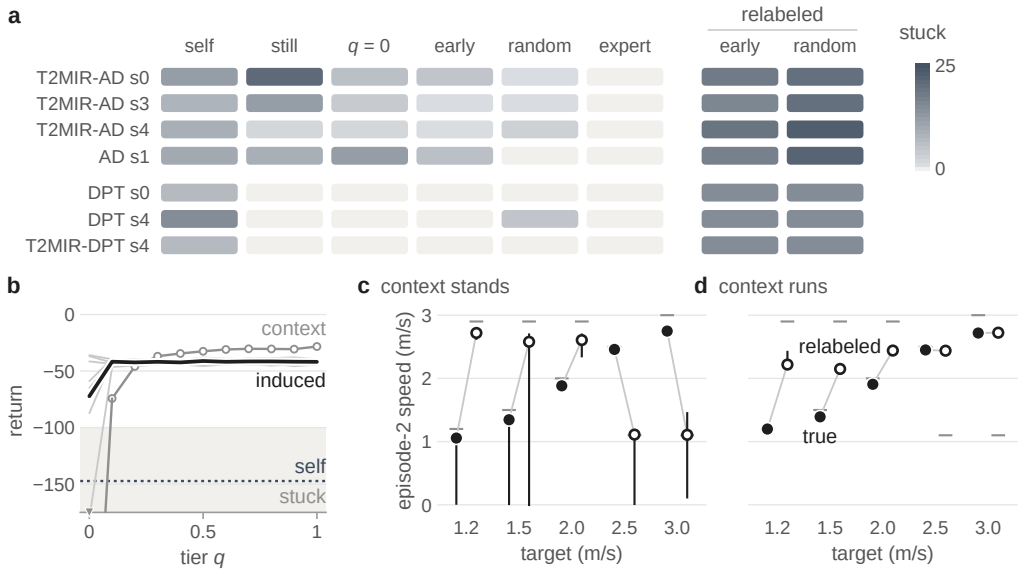


Figure 16: **All context swaps** (checkpoints and resets as in Figure 3). (a) Stuck trials of 25 per checkpoint and context episode, darker for more; palest, none. *early*: an early offline episode of the held-out task's own SAC run (mean return -280); *relabeled*: the early or the random episode with rewards for a wrong target θ'. None of the 68 self-stuck trials stands under a relabeled arm: they run at the wrong speed. (b) Last-5 return induced by a context episode taken from the chronological quantile q of the held-out task's SAC run, per checkpoint (thin) and on average (bold), with that episode's own return (gray; the triangle marks $q = 0$, at -416); dotted, the self context; band, stuck. Every tier $q \geq 0.1$ leaves 0 of 25 trials stuck on each of the seven checkpoints. Neither lineage trains on checkpoint order (Appendix B), and these episodes carry held-out-task rewards a deployed agent would not have. (c, d) As Figure 3b for the early episode, for the 105 trials whose context episode stands (c, 21 per target) and the 70 whose context episode runs (d, 14 per target). Every early trial that ends stuck had a standing context, and 113 of 175 relabeled trials run closer to θ' than to θ.

deviation 0.17, against a pretraining minimum of 0.245, with 164 of these 165 standing starts below it). Not drawn in the panel: all 165 also have a lower mean joint speed, the mean absolute joint velocity over the episode, than every pretraining episode. Over the 168 stuck trials that the rerun matches to the grid, 164 have a lower mean per-step speed $\frac{1}{H}\sum_t |v_t|$ than every pretraining episode (Section 4). This reflects deterministic decoding, not the stuck mode itself: 161 of the 185 standing starts that escape also vary their actions less than every pretraining episode. Per step (b), we measure the mean distance of each (state, action) pair to its five nearest pretraining pairs and calibrate a threshold τ_{95} on held-out-task data. Standing self episodes lie close to the earliest tenth of the pretraining data: 1.4% of their steps fall beyond τ_{95}, and the nearest tier is the first for 94% of these episodes, where a tier is one chronological decile of a training task's episodes. Early SAC episodes also stand on average, which explains the proximity: 69% of the first five episodes of each training task have a mean speed below 0.1 m/s in absolute value. No pretraining episode has a mean per-step speed $\frac{1}{H}\sum_t |v_t|$ below 0.1 m/s, and the minimum is 0.21 m/s. Still, 255 of the 7,410 episodes, 253 of them in the first tier, have a mean forward speed below 0.1 m/s in absolute value, so they stand on average. A uniformly random episode is far: 96.4% of its steps lie beyond τ_{95} (90.6% for states alone), and 99.9% lie beyond the corresponding threshold when the reference is only the first five episodes of every training task. Distance does not predict the outcome (c). Within a checkpoint, the nearest-neighbor distance of a standing start does not separate those that stay stuck from those that escape (AUROC 0.52, 95% interval $[0.41, 0.60]$). The checkpoint explains 29% of the variance of the outcome and the (checkpoint, task) cell 57%. So the standing AD-lineage episode is off the pretraining data in its per-step stillness and action variability rather than in the states it visits, and a random episode is off the data everywhere. Coverage (Lemma 2) explains neither which trials stay stuck nor why a random episode helps.

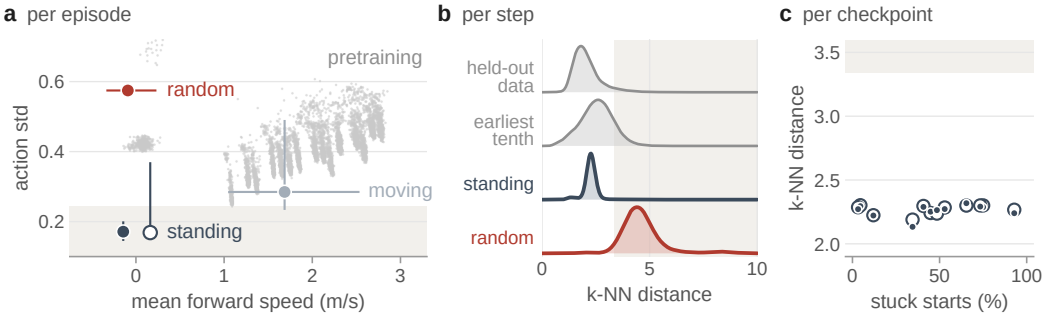


Figure 17: **Where the first episode lies relative to the pretraining data** (HalfCheetah-Vel, episode 1 of held-out trials on the 24 checkpoints with trajectory logs; stuck labels from the logged reruns, 169 AD-lineage and 102 DPT-lineage stuck trials). (a) Within-episode action standard deviation against mean forward speed. Gray: all 7,410 pretraining episodes; the cluster near zero speed is the earliest tier, which also stands on average. Shaded: below the lowest pretraining value, 0.245. Markers are group medians of first episodes, whiskers span the 5th to 95th percentile. Standing self starts (mean speed below 0.1 m/s) that stay stuck (165, filled) and that escape (185, hollow) lie at the same spot, below every pretraining episode (164 of 165 and 161 of 185). Moving starts (850, among them all 600 DPT-lineage trials, none of which starts standing) lie inside the cloud; the 395 distinct random episodes lie above it. (b) Per-step distance to the 5 nearest pretraining (state, action) pairs, one density per group scaled to its own peak; shaded, beyond τ_{95}, the 95th percentile on held-out-task data (5.0% of those steps by construction). Share of steps beyond τ_{95}: earliest tenth of the held-out-task data 13.6%, standing self 1.4%, random 96.4% (90.6% beyond a state-only threshold). (c) Per AD-lineage checkpoint, the median distance of the standing starts that stay stuck (dot) and that escape (ring) against the share of standing starts that stay stuck; shaded, beyond τ_{95}. The two distances agree to within 0.06 on every checkpoint (within-checkpoint AUROC 0.52, 95% interval $[0.41, 0.60]$) while the share runs from 3 to 93%.

F.5 Activation steering

Question and design. If the frozen model encodes the task parameter in its activations, adding the right direction might supply the task information that the self-written context fails to elicit, without changing the context, in the spirit of task vectors in language models (Hendel et al., 2023). We tested this on T2MIR-AD seed 0. For each training task k and hook l, the task vector $v_k(l)$ is the mean residual-stream activation over the context's state tokens at l, computed with an expert context of task k and 32 shared query states. At deployment the hook adds $\alpha\, v(l)$. On training tasks it uses the vector of the true task, and on held-out tasks a vector interpolated between the two nearest training tasks. A hook that adds zero reproduces the self-context baseline bit for bit. Before running the closed loop we fixed a pass bar: with the vector of the true training task, at most 2 of 25 trials stuck and a last-5 return within 10% of the expert-context ceiling.

Is there a linear task direction? A leave-one-out ridge regression of the task parameter on $v_k(l)$ over the training tasks reaches $R^2 = 0.965$ on HalfCheetah-Vel (embedding layer) and 0.817 on Hopper-Friction (input of layer 1). On MetaWorld Reach it stays below 0.10 for every goal coordinate at every hook. Reach uses 12 of its 15 training tasks, because three of its SAC experts failed (episode returns of 102 to 127, against 1,400 to 1,580 on the other tasks). The expert contexts of those tasks carry no goal information. The frozen model thus encodes a scalar task parameter linearly, but not a hidden three-dimensional goal.

Closed-loop result. Steering has a real causal effect but falls far short of the bar (Table 13). The best configuration on training tasks, layer 2 with $\alpha = 1$, raises the last-5 return from -172.2 to -95.7, which recovers about half of the gap to the ceiling of -26.9, and leaves the stuck count at 10 of 25. No configuration brings it below 8 of 25. The effect is not monotone in α: doubling the scale at layer 1 undoes the gain and raises the stuck count to 20 of 25. Per trial, the vector moves the return by several hundred in the right direction on average, yet it does not land on the target speed. On one task it rescues all stuck trials, and on another it lifts the stuck trials while pulling the good trials down to the same level. On held-out tasks the interpolated vector transfers: it improves the last-5 return by about 100 at either layer, again about half of the gap to -32.9. Again neither layer reduces the stuck count below 10 of 25. On Hopper-Friction, which has no stuck mode, steering at

Table 13: **Activation steering of T2MIR-AD seed 0 on HalfCheetah-Vel.** 5 tasks × 5 trials × 10 episodes per row. L1 and L2 denote the inputs of layers 1 and 2, and α scales the added vector. Training tasks use the vector of the true task, held-out tasks a vector interpolated between the two nearest training tasks.

Deployment	episode 1	episode 10	last-5	stuck
Training tasks, true-task vector				
expert context (ceiling)	−25.1	−30.2	−26.9	0/25
self context	−184.3	−167.9	−172.2	10/25
steer embedding, $\alpha = 1$	−280.2	−177.8	−179.7	16/25
steer L1, $\alpha = 0.5$	−202.4	−118.4	−133.1	8/25
steer L1, $\alpha = 1$	−226.8	−98.1	−103.2	10/25
steer L1, $\alpha = 2$	−257.7	−167.2	−168.5	20/25
steer L1, $\alpha = 1$, state tokens only	−247.7	−216.2	−222.3	16/25
steer L1, $\alpha = 1$, query token only	−247.2	−204.0	−224.3	17/25
steer L2, $\alpha = 1$	−187.3	−94.6	−95.7	10/25
Held-out tasks, interpolated vector				
expert context (ceiling)	−32.7	−33.2	−32.9	0/25
self context	−249.4	−236.2	−236.0	12/25
steer L1, $\alpha = 1$	−248.8	−129.4	−133.0	10/25
steer L2, $\alpha = 1$	−201.4	−111.1	−136.5	12/25

layer 1 leaves episode 1 essentially unchanged (72.9 against 78.7) and moves the last-5 return from 598.6 to 619.2, both within the noise of five trials.

What it shows. Since even supplying the true-task direction recovers only half of the gap, no scheme of this kind (adding a mean-activation task vector at these hooks) could reach the ceiling on this checkpoint. On MetaWorld, whose goals the frozen model does not encode linearly, the mechanism is unavailable. We therefore do not use steering as a method. The study does support the reading of Section 4: on HalfCheetah-Vel the frozen model represents the task parameter, and the failure lies in what the self-written context elicits.

F.6 Group-relative policy-gradient post-training

Question and design. If the attractor is a property of the deterministic policy on its own contexts, fine-tuning the model on its own trials with a trial-level reward might remove it. We post-trained T2MIR-AD seed 0 with a group-relative policy-gradient objective (Shao et al., 2024). Each update rolls out 5 training tasks × 8 trials under the exact deployment rule of the model (four-episode buffer sorted by return, the lowest-return episode of the previous buffer dropped at overflow, absorbing padding), with actions $a = \mathrm{clip}(\mu_\phi(s, C) + \sigma\varepsilon, -1, 1)$. The 8 trials of a task share one environment seed sequence, so within a group only the exploration noise differs. The trial reward is the last-5 return. The advantage is the reward minus the group mean, divided by the group standard deviation plus 10 return units, a floor on the denominator (trial returns are of order hundreds). The loss is a clipped importance-ratio surrogate (clip 0.2) on the Gaussian log-probability of every action in every episode of the trial, plus a KL penalty to the frozen checkpoint with weight 0.02. The surrogate covers the whole trial, so early episodes are credited for the context they build. We used AdamW with learning rate 5×10^{-6} and gradient-norm clipping at 1.0, and trained on training tasks only, with environment seeds disjoint from the evaluation seeds. Two runs with $\sigma = 0.1$ and $\sigma = 0.3$ each took 150 updates (6,000 trials, 12 million environment steps), and together they used about 29 GPU hours. The pass bar, fixed in advance, was at most 2 of 25 held-out trials stuck and a held-out last-5 return of at least −53.

Result. Optimization was stable throughout (KL to the checkpoint 0.07 to 0.15, clipped fraction 1% to 3%), but neither run approached the bar (Table 14). In the periodic held-out evaluation of the deterministic policy, the $\sigma = 0.1$ run lowered the stuck rate from 0.48 to 0.32 by update 40 and then fluctuated between 0.24 and 0.36 with no trend, while the $\sigma = 0.3$ run stayed between 0.40 and 0.48. Under its own stochastic deployment the post-trained model improves on the same noise applied to the base checkpoint (−124 against −178), so the post-training did change the policy in the intended direction. The deterministic policy, however, stayed far from the bar (−170 against −236, 8 of 25 trials stuck), and plain test-time noise with $\sigma = 0.3$ on the base checkpoint (−96) beats the post-trained model under either deployment.

Table 14: **Policy-gradient post-training of T2MIR-AD seed 0 on HalfCheetah-Vel.** Held-out tasks, 5 tasks $\times$ 5 trials $\times$ 10 episodes, scored with the same closed-loop script as Table 12. Test-time noise is Gaussian with the given σ in every episode.

Deployment	last-5	stuck
expert context (ceiling)	-33	0/25
base checkpoint, deterministic	-236	12/25
base checkpoint, test-time noise $\sigma = 0.1$	-178	8/25
base checkpoint, test-time noise $\sigma = 0.3$	-96	9/25
post-trained ($\sigma = 0.1$ run, update 80), deterministic	-170	8/25
post-trained ($\sigma = 0.1$ run, update 80), test-time noise $\sigma = 0.1$	-124	6/25

What it shows. On this checkpoint, the stuck mode did not yield to this post-training of the deterministic policy, while a random first episode released the stuck probe trials (Appendix F.1). Plain test-time noise did better than post-training here. In the grid, per-step noise matches a random first episode in the AD lineage on mean last-5 return, but it makes a third of the non-stuck HalfCheetah-Vel trials stuck (Appendix E). These results are one reason we pursued a training-free remedy that changes the context directly.

G PROMPT-DROPOUT FINE-TUNING DETAILS

Procedure. We start from the 100k-step pretrained checkpoint of T2MIR-DPT or DPT. We continue training on the same offline data with the method's own objective: the action-prediction loss, and for T2MIR-DPT the auxiliary load-balancing and contrastive losses of its mixture-of-experts layers. The only change is the prompt mask. For each sequence in a batch, with probability p we set the prompt positions of the attention mask to zero, so the model must predict the query action from the query alone. We use $p = 0.5$ for the first 10k steps and $p = 0.2$ afterward. The fine-tune uses AdamW with learning rate 10^{-4}, weight decay 10^{-4}, batch size 32, a linear warm-up over 2,000 steps followed by cosine decay, and gradient-norm clipping at 1.0. A 20k-step fine-tune takes 0.4 to 0.9 h on one A100. Evaluation uses the unchanged harness and deployment rule: an empty prompt at episode 1, then the model's own previous episode as the prompt.

Controls. In the grid, the control continues training each checkpoint for the same 20k steps, with the same optimizer, schedule and data, but no masking. We run it on all ten families, six seeds of each DPT-lineage method. Each arm, dropout and the step-matched control, has one fine-tuning run per checkpoint.

Single checkpoints and training length. In a HalfCheetah-Vel pilot on DPT seed 0 and T2MIR-DPT seed 4, dropout left at most one of 50 trials stuck after 10k steps and none after 20k or 30k, with the same last-5 return at 20k and 30k within noise; we use 20k steps. Continued training moved the stuck count of T2MIR-DPT seed 4 from 12 of 50 to 4 in a 30k-step pilot run but to 19 in its 20k-step grid run, so single-checkpoint controls are noisy, and the comparison in Section 7.3 rests on the twelve grid checkpoints.

The grid. We pair every difference below over the twelve checkpoints and report mean $\pm$ s.e. The grid fine-tunes T2MIR-DPT and DPT on all ten families with six seeds each, 120 runs of 20k steps, and evaluates each run under self and under `always3`. Figure 18 gives the per-family results. In last-5 return, dropout wins on HalfCheetah-Vel and Push and ties on six families. Dropout loses on Hopper-Mass (-5.7 ± 2.5, with a 95% interval below zero) and Soccer (-98.9 ± 75.5). Against the step-matched control the difference is -1.2 ± 2.3 on Hopper-Mass and -70.8 ± 89.7 on Soccer, so neither loss is attributable to the masking. EP wins on eight families. Dropout costs no episode, and its AUC is higher than EP's on all ten families at one standard error (on six at 95%). Dropout lowers the episode-1 return (-161.3 against -146.2 on HalfCheetah-Vel, lower on 9 of 12 checkpoints). Continued training changes little on HalfCheetah-Vel ($+1.7 \pm 5.9$), Reach ($+7.4 \pm 4.6$) and Pick-Place ($+33.9 \pm 41.7$, where dropout minus continued training is -13.0 ± 32.4). Continued training gains on Push ($+258.8 \pm 128.8$), where dropout gains more ($+452.7 \pm 160.3$). On the other six families it moves the last-5 return by -4.5 ± 2.9 (Hopper-Mass), $+3.3 \pm 4.5$ (Hopper-Friction), $+3.9 \pm 8.5$ (Walker-Mass), 0.0 ± 3.1 (Walker-Friction), $+6.8 \pm 9.7$ (Humanoid-Dir) and -28.2 ± 31.7 (Soccer). On HalfCheetah-Vel, 100 of 600 trials are stuck under self, 95 after continued training, 19 after dropout, 0 under EP and 3 under EP + dropout. Under dropout, success within 0.10 m moves

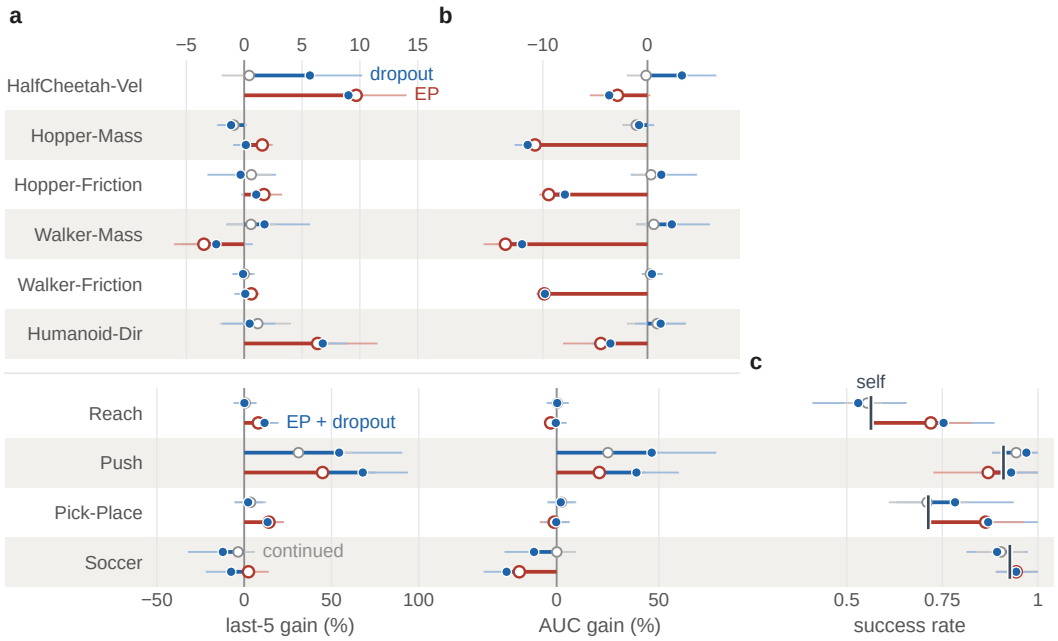


Figure 18: **Prompt dropout on all ten families** (DPT lineage, 12 checkpoints, paired differences over checkpoints). (a) Last-5 gain over self as a percentage of the random-to-demo gap; upper line of each family, self context (blue dot and bar from 0, dropout; gray ring, continued training); lower line, EP context (red ring and bar from 0, EP; blue dot and bar from EP, EP + dropout); whiskers, 95% hierarchical-bootstrap interval of the paired difference that each marker represents (for EP + dropout, its difference from EP), drawn around that marker. The MuJoCo and MetaWorld blocks have separate scales. (b) The same for AUC. Dropout has a higher AUC than EP on all ten families at one standard error. (c) MetaWorld success within 0.10 m; slate tick, self context. The demo return is a scale, not a ceiling: EP exceeds it on HalfCheetah-Vel (-38.7 against -50.3), and EP + dropout reaches 1322.5 on Push against 844.5.

from 0.56 to 0.53 on Reach, from 0.91 to 0.97 on Push and from 0.71 to 0.78 on Pick-Place. EP + dropout beats EP alone on Reach ($+44.8 \pm 44.1$) and Push ($+191.6 \pm 94.5$). EP + dropout loses to EP alone on HalfCheetah-Vel, Hopper-Mass, Walker-Friction and Soccer at one standard error, so the two remedies are not additive.

Interpretation. One reading of the dropout effect is that training on masked prompts gives the decoder a prompt-free action distribution for each state, so that a poor self-written prompt pulls the policy less strongly toward reproducing itself. These controls do not establish the mechanism by which masking during fine-tuning changes how the model reads later, nonempty prompts.

H Reproducibility

This appendix says what a reader can check, and how a number in a table can be traced back to the episodes that produced it.

H.1 What is released

The supplementary material contains the code for exploratory prompting: the context rule for both lineages as stated in Appendix E, the prompt-dropout mask schedule of Appendix G, a toy task family on which an entry point runs the rules of both lineages, and a smoke test. It contains no checkpoints, offline data, simulator environments or evaluation harness; each baseline is trained and deployed with its own released code (Appendix B).

H.2 How a number can be traced

Scoring goes through one harness. Across methods it shares the environment, the cold start with an empty context, the number and length of episodes, the return and success bookkeeping, and the environment seed of every (task, trial). It deliberately does not share the context rule: each method has its own adapter, which reproduces the deployment logic of its release. The prompt-dropout checkpoints are scored by the unchanged DPT-lineage adapter. Because each release expects its own task numbering, the map from each release's task index to the absolute task and its parameter is written once and read by both training and evaluation. A wrong renumbering would train a model to predict one task's actions from another task's context without any visible error.

Every evaluation writes one record per (task, trial, episode) with the family, method, seed, context rule, absolute task index, trial, episode, return, success and step count. MetaWorld records also carry the minimum distance to the goal. A cell is identified by its family, method, seed and context rule, and the merged records are arranged by those keys. The environment of each (task, trial) is seeded with the base seed 20260905 plus 1000 times the task index plus the trial index. That seed is applied to the inner MuJoCo environment because the wrapper's own seeding call has no effect. Every method therefore faces the same initial states.

One script turns these records into the per-cell statistics of the main text. For each cell it computes the per-trial last-5 return, the ten-episode AUC and the HalfCheetah-Vel stuck indicator (last-5 below -100). On MetaWorld it also computes success@0.10 from the minimum goal distance and success@flag from the environment's success flag. It then averages over (task, trial) rows to one value per seed, and reports mean $\pm$ std over seeds. The paired verdicts compare a rule cell with the self-context cell of the same family, method and seed. The script writes a human-readable ground-truth record and the LaTeX body of Table 2. That body is inlined verbatim in this paper, so a table cell is a transcription of the script's output rather than a second computation. The rule comparisons of Figure 6 and Table 10 are produced by a second script from the same records with the same pairing. The single-checkpoint probes of Appendix F and the prompt-dropout pilot of Appendix G run through the same harness loop and write per-episode returns together with a summary of the mean curve, last-5 return and stuck count. The numbers printed for them are copied from those summaries into the same ground-truth record. Every figure is drawn by a script that also logs the numbers it plots in a file next to the figure. The analyses in Appendices A, B, C, E and F that produce no figure (repeat rates, pretraining tiers, counts of trials released and newly made stuck, commit rates, compute) are computed by scripts over the same records.

I Extended related work

This appendix extends Section 2 with the details of each line of work and how our results relate to it.

In-context reinforcement learners. The Decision Transformer and the Trajectory Transformer cast offline reinforcement learning as sequence modeling (Chen et al., 2021; Janner et al., 2021). Algorithm Distillation (AD) trains a transformer on across-episode learning histories (Laskin et al., 2023). AD expects the frozen model to improve as its own episodes fill the context. Later work builds such histories by noise injection (Zisman et al., 2024), filters them for steady improvement (Chen et al., 2025), adds model-based planning (Son et al., 2025), changes the backbone (Beaussant & Mounsif, 2025), or scales the recipe across domains (Polubarov et al., 2025). The Decision-Pretrained Transformer (DPT) instead predicts the optimal action from an in-context dataset (Lee et al., 2023). Its follow-ups do without optimal-action labels: they distill labels from random-policy rollouts within a trust horizon (Chen & Paternain, 2025) or weight suboptimal in-context data by an estimated advantage (Dong et al., 2025). Polubarov et al. (2026) keep demonstrator action labels and scale DPT across domains with a flow-matching action head. Related agents condition on a trajectory prompt (Xu et al., 2022), world-model task codes plus a prompt chosen by world-model error (Wang et al., 2024) or a world-model prompt while learning in-context value functions (Liu et al., 2026), or add mixture-of-experts routing (Wu et al., 2025). Lin et al. (2024) show that pretrained transformers can implement reinforcement learning algorithms in context, with an error term that grows with the ratio between the trajectory distributions of the expert algorithm and of the offline algorithm that generated the pretraining data. Moeini et al. (2025) survey the area. These

methods differ in whether evaluation starts from an empty context or from a demonstration of the held-out task. We evaluate seven of them from an empty context under one harness (Section 3).

Failures of self-generated context. Prior work has observed that a pretrained in-context learner can be misled by its own history. Zhang et al. (2025) call this contextual ambiguity. In gridworlds, stochastic action sampling within an episode makes the model misread which stage of learning its context represents. Their remedy disambiguates the context by adding an estimated context value to the model's input at every step. Language models that learn in context from reward show suffix failures, in which the best arm is never chosen after an initial phase (Krishnamurthy et al., 2024). They are also sensitive to how their own past outputs are kept in context (Monea et al., 2025; Song et al., 2026). At scale, Polubarov et al. (2026) report that, on held-out tasks, cold-start evaluation is on par with demonstration-prompted evaluation on eight of their ten domains but trails it on the two manipulation benchmarks, and suggest that test-time exploration remains limited. Beaussant & Mounsif (2025) report that on HalfCheetah-Vel their Mamba-based AD plateaus after a zero-shot attempt unless its context spans the whole subsampled learning history. Offline meta-RL describes related problems. Context shift arises when the policy that produced the training contexts differs from the one that collects the test contexts (Gao et al., 2023; Pong et al., 2022). MDP ambiguity arises when offline data do not identify the task (Dorfman et al., 2021). Sequence-model imitation shows latching and copycat behavior, in which the policy follows its own past actions (Swamy et al., 2022; Wen et al., 2020), a relative of compounding error on the learner's own state distribution (Ross et al., 2011). We isolate a distinct trigger that does not require sampling noise and that accounts for the stuck AD- and DPT-lineage trials we observe on HalfCheetah-Vel. With the deterministic action heads used by continuous-control ICRL baselines, the empty-context policy produces one behavior per checkpoint. On HalfCheetah-Vel that first episode can be atypical of the pretraining data: AD-lineage agents stand still and DPT-lineage agents overshoot. The context the agent then writes for itself repeats or amplifies that episode although its rewards carry the task (Section 4). Where it repeats, it approximates a fixed point in the sense of Lemma 1. This is the transformer-ICRL analogue of context shift in offline meta-RL and of latching in sequence-model imitation. Unlike contextual ambiguity, it requires no sampling noise. Our per-step noise control, which hurts the DPT lineage, is consistent with the account that Zhang et al. (2025) give for AD-like models.

Exploration in meta-RL. RL^2 trains a recurrent policy across the episodes of one task, so early episodes can be spent on exploration (Duan et al., 2016). E-RL^2 removes the return of exploration episodes from the objective (Stadie et al., 2018). DREAM learns a separate exploration policy whose trajectories recover the task-relevant information (Liu et al., 2021). VariBAD and HyperX explore under a learned task belief, the latter with exploration bonuses during meta-training (Zintgraf et al., 2020; 2021). MetaCURE rewards exploration by task-inference information (Zhang et al., 2021), PEARL samples one task hypothesis per episode (Rakelly et al., 2019), and ICEE trains a transformer to trade off exploration and exploitation in context (Dai et al., 2024). Active system identification chooses actions that make physical parameters identifiable (Memmel et al., 2024). See Beck et al. (2025) for a survey. All of these learn exploration or task inference during training. Exploratory prompting revives the separate exploration episode of E-RL^2 for frozen pretrained transformers. It trains nothing, and its exploratory episode is uniformly random, the simplest choice. A Gaussian or ε-greedy episode also helps, though less on HalfCheetah-Vel (Appendix E). A random episode is not close to the pretraining data (Appendix F.4), and we do not claim that it maximizes information. That the frozen model reads its rewards is an empirical finding (Figure 3).

Prompt tuning and selection. Several methods improve a decision transformer by changing the trajectory segments it conditions on. Hu et al. (2026) tune prompt tokens with black-box perturbations ranked by return or preference feedback, and Prompt Diffuser generates prompts with a return-conditioned diffusion model whose training is guided by the pretrained model's action loss on target-task data (Hu et al., 2024). Meta-DT selects the history segment on which its world model errs most (Wang et al., 2024), and RA-DT retrieves sub-trajectories from an external memory by embedding similarity and reads them through cross-attention layers (Schmied et al., 2026). Closest to our rule, Rietz et al. (2025) treat prompt tuning as a bandit. Its arms are segments of expert demonstrations, its reward is the return a prompt induces online, and the authors prove a regret bound. Our prompt selection is a demonstration-free instance of bandit prompt tuning. Its candidates are the agent's own first episode and one exploratory episode, and we apply it at the cold start, where no demonstration exists. The comparison step is a two-armed explore-then-commit rule (Lattimore & Szepesvári, 2020), and our selection bound is the standard one for that rule (Proposition 4). The

candidate set matters: Xu et al. (2022) report that a low-quality prompt, taken from the first trajectories of an RL agent's replay buffer, lowers the return of a Prompt-DT trained on expert data and prompts. Our models read a random episode although it is far from their pretraining data (Appendix F.4). Whether models trained only on expert prompts would do the same is open. Prompt-DT is one of the methods to which we do not apply EP (Section 5).

Conditioning dropout. Dropping the conditioning signal during training gives one network both a conditional and an unconditional mode. Classifier-free guidance does so for diffusion models (Ho & Salimans, 2022), Decision Diffuser for return and constraint conditioning in offline RL (Ajay et al., 2023), and CFGRL for optimality conditioning of policies (Frans et al., 2025). Octo instead drops one of its two task inputs, goal image or language instruction, so that one generalist robot policy accepts either (Octo Model Team et al., 2024). Prompt dropout is conditioning dropout applied to the in-context history. In our controls it does not improve the empty-context first episode. It changes how the model continues from its own weak episodes, and that change cuts HalfCheetah-Vel stuck trials from 100 to 19 of 600. Step-matched continued training leaves 95 (Section 7.3). Prompt dropout requires no change at test time, and we use no guidance weight.

Stochastic action heads. Online Decision Transformer learns a stochastic policy with an entropy constraint so that online fine-tuning explores (Zheng et al., 2022). DPT's online protocol samples actions from the predicted distribution, which its authors relate to posterior sampling (Lee et al., 2023; Osband et al., 2013). In continuous control, ICRL methods commonly act with the mean action at test time (Xu et al., 2022; Wang et al., 2024; Wu et al., 2025; Polubarov et al., 2025), and our harness does so for all seven baselines. Per-step action noise matches a random first episode in the AD lineage on mean last-5 return, but it makes a third of the non-stuck HalfCheetah-Vel trials stuck. In the DPT lineage it loses 11 of 20 pairs, and a constant per-episode action offset hurts as well (Section 7, Appendices E and F.1). In the DPT lineage, and on HalfCheetah-Vel in both lineages, what matters is which episode enters the context rather than whether actions are perturbed.