DYNAMICS GENERALIZATION WITHOUT ADAPTATION MACHINERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned dynamics models typically fail to generalize to environments with unseen physical parameters. Standard approaches add adaptation machinery, such as test-time gradient updates or an inferred latent variable trained with an auxiliary objective. We show that this machinery is not what decides the matter, and that whether a model adapts in context is settled by how it is trained and by where it is tested. A plain sequence model, with no module for inferring the environment, no loss beyond next-state prediction and no positional encoding, adapts to unseen physics in a single forward pass with its weights frozen. Its context is simply the recent transitions, each kept whole so that a state, the action taken, and the resulting next state stay together. On the six continuous-control environments of Lee et al. (2020), under their protocol and interaction budget, it outperforms the vanilla, stacked, and meta-learned model-based baselines everywhere, and an ensemble variant clears both variants of the purpose-built context-aware method outright on three of the six while staying inside seed noise on two more. Interventions that leave the weights untouched show what the model uses. Permuting which next state follows which transition costs almost as much as removing the context outright, and replacing the transitions by their average costs two-thirds of the benefit, so the model addresses individual transitions rather than a summary of them. What the context is worth also depends on where the model is tested, moving by a factor of five within one environment between held-out ranges that sit equally far from training. We report where the approach does not work. On Ant, the context is worth nothing, and no intervention we tried changes that, including a fivefold widening of the training distribution.

1 INTRODUCTION

A learned dynamics model is tied to the physics it was trained on. Changing the mass of a robot, the damping of its joints, or disabling an actuator makes the model's predictions drift. Model-based control then degrades even when the planner is unchanged. This is the dynamics-generalization problem studied by Lee et al. (2020), and it is the setting of this paper. There exist three families of remedies. Meta-learning adapts the model with a few gradient steps on fresh data (Nagabandi et al., 2019). Context encoders learn a latent variable that summarizes recent transitions, trained with an auxiliary objective that predicts forward and backward dynamics (Lee et al., 2020). Ensembles average over epistemic uncertainty instead of adapting (Chua et al., 2018). Each adds machinery to the training pipeline and assumes the adaptation signal must be engineered in.

We take the opposite route and ask what a sequence model does on its own. Our model reads a short window of recent transitions as context and predicts the next state. The model receives no explicit environment identifier, uses no auxiliary loss to shape the context representation, and takes no gradient steps after deployment. Adaptation happens inside the forward pass, in the sense of in-context learning (Wang et al., 2025; Cole et al., 2024). However, the question is whether that is enough for control.

Our evidence shows that it is enough to pass the vanilla, stacked and meta-learned model-based baselines everywhere, and enough for an ensemble to clear both variants of the purpose-built context-aware method on three of the six environments and stay inside seed noise on two more. We evaluate on all six environments of the CaDM benchmark with the original protocol, model-selection rule and interaction budget. We report every published row rather than a favorable subset. That includes

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.

the model-free rows, which use between 25 and 125 times our interaction budget and beat every model-based method on three environments.

Contributions. Our contributions are outlined as follows.

- First, we demonstrate that in-context adaptation of a dynamics model is sufficient for zero-shot generalization and control under changing dynamics. A single model trained with forward-pass adaptation alone outperforms vanilla dynamics models, stacked baselines, and both meta-learned adaptive methods across every benchmark environment and evaluation regime, all without test-time gradient updates.
- Second, we isolate the underlying mechanism. Under frozen weights, removing context at test time drops returns by 1104 on HalfCheetah and 362 on Pendulum, with just two transitions restoring half of the loss. Scrambling transition pairs or substituting them with matching Gaussian noise degrades performance nearly as much as full context deletion, while averaging transitions forfeits two-thirds of the gain. These ablations confirm that the Transformer extracts fine-grained dynamics from individual transitions, rather than relying on sequence length or coarse distributional summaries.
- Lastly, we show that context value is governed by the direction of distribution shift, not the environment itself. Within a single environment, context utility varies by a factor of five across test ranges equidistant from the training distribution, which is an effect we reproduce under frozen weights by transferring range geometry across domains. This insight provides a unifying lens for our two edge cases: on HalfCheetah, the context-aware baseline yields its largest gains in the extreme regime by suppressing out-of-distribution prediction error, whereas on Ant, the lack of context gain reflects its asymmetric, one-sided evaluation protocol. Because our architecture remains identical across all six environments, these results demonstrate that shift geometry rather than model capacity determines context efficacy.

2 Related work

Adapting to changing dynamics. Table 1 sets out what each family adds. Domain randomization trains across randomized physical parameters and relies on robustness rather than adaptation (Peng et al., 2018), sometimes with an explicit online parameter estimate (Yu et al., 2017). Meta-learning recovers accuracy with a few gradient steps on recent data, as in GrBAL and ReBAL (Nagabandi et al., 2019; Beck et al., 2023). Context encoders learn a latent variable from recent transitions and feed it to the model; CaDM trains that encoder with forward and backward prediction losses and is the strongest published method on this benchmark (Lee et al., 2020). Probabilistic ensembles average over epistemic uncertainty instead of adapting, which PETS combines with trajectory sampling (Chua et al., 2018). Our model has none of these parts, in particular no estimate of the parameters, explicit or latent.

In-context learning. Sequence models adapt to a task from examples in context, without weight updates (Zhao et al., 2026; Chen et al., 2025). Garg et al. (2022) showed that a transformer trained from scratch on a function class recovers an estimator competitive with least squares, and von Oswald et al. (2023) relate what attention computes on a prompt to a step of gradient descent on that data. In reinforcement learning, Laskin et al. (2023) distil a learning algorithm into a causal sequence model that improves its policy in context. Our setting differs in what the context holds: not input-output examples of a function, and not an episode of a learning history, but the transitions of the system being controlled, so what the model must recover is the dynamics. Recent work studies this for world models, asking when a model recognizes a known environment and when it infers an unseen one from the context alone (Wang et al., 2025; Cole et al., 2024; Schaeffer et al., 2024; Prasanna et al., 2024). Related studies examine in-context world models for prediction and language-conditioned settings (Wu et al., 2025; Nguyen & Lee, 2025); they measure prediction. We ask whether the same capability controls a physical system whose parameters change, under a published benchmark protocol and its interaction budget.

World models for control. Learned simulators support planning and policy learning in imagination (Hafner et al., 2019; Hansen et al., 2023), and the surveys of Luo et al. (2024); Moerland et al. (2023); Ding et al. (2025) cover the range of designs; Quevedo et al. (2025) evaluates policies inside one. Most of this work assumes a fixed environment distribution at test time (Zhou et al., 2024). Our setting keeps the planner fixed and changes only what the model reads, isolating adaptation

Table 1: What each method adds in order to adapt. *Module* is a component whose only job is to represent the environment. *Extra loss* is a training objective beyond predicting the next state. *Test-time gradient* is whether weights change after deployment. The last column is the cost of one adaptation. Our model is the only row that adds nothing.

Method	Module	Extra loss	Test-time gradient	Reads context	Cost of one adaptation
Vanilla DM (Lee et al., 2020)	✗	✗	✗	✗	—
Stacked DM (Lee et al., 2020)	✗	✗	✗	✓	one forward pass
PE-TS (Chua et al., 2018)	✗	✗	✗	✗	—
GrBAL, ReBAL (Nagabandi et al., 2019)	✗	meta-objective	✓	✓	gradient steps
CaDM (Lee et al., 2020)	encoder	backward model	✗	✓	encoder pass
Ours	✗	✗	✗	✓	one forward pass

from planning; Roice et al. (2024) makes a related separation between what a planner searches over and what a model supplies, as in Dyna-style planning (Sutton et al., 2012). Ensembles enter our setting only as the uncertainty mechanism of Chua et al. (2018), and the disagreement they measure serves elsewhere as an exploration signal (Sekar et al., 2020; Wahab et al., 2026), not for adaptation.

3 Problem setting

Environment family. We consider a family of Markov decision processes that share a state space $\mathcal{S}$, an action space $\mathcal{A}$ and a reward function r, and differ in their transition kernel. A parameter vector ψ fixes the physics, for example the mass scale of a robot, the damping of its joints, or which actuator is disabled. Training draws ψ from a set Ψ_{train} at the start of every episode. Testing draws ψ from disjoint sets that lie outside the training range, in two regimes. The moderate regime sits next to the training range and the extreme regime sits further out.

Adaptation without weight updates. The agent is not told ψ and cannot estimate it offline. It observes a stream of transitions and must adapt within the episode. We write the last K transitions:

$$C_t = \big((s_{t-K}, a_{t-K}, s_{t-K+1}), \ldots, (s_{t-1}, a_{t-1}, s_t)\big), \tag{1}$$

and ask the model for the next state given the current state, a candidate action and that context,

$$\hat{s}_{t+1} = f_\theta(s_t, a_t \mid C_t). \tag{2}$$

The weights θ are trained once on Ψ_{train} and never updated afterwards. All adaptation happens inside the forward pass of equation (2).

4 Method

The method is deliberately simple: a sequence model predicts the next state from recent transitions, and a planner uses those predictions (figure 1). No additional components are learned, and no parameters are updated after deployment.

4.1 Model

Transitions as tokens. Each context item represents a single transition, encoded as a tuple of the current state, the action taken, and the resulting next state. The query token contains the current state paired with a candidate action. Using self- and cross-attention, the Transformer attends from the query to both the context items and itself. A downstream prediction head then outputs the predicted state difference as a mean and log-variance. Both states and state differences are normalized using dataset statistics.

Keeping the next state inside the item is the one design choice we insist on. The map from a state-action to its successor is what identifies the physics. A pair that holds only a state and an action is consistent with every parameter setting in the family, so it carries no information about which one is active. In the empirical evaluation, we test this by permuting next states across the context, and also by training a model whose items are pairs.

Reading the context costs one forward pass. Because the context is encoded only once per decision, adaptation requires no gradient computation at test time. This zero-overhead step represents

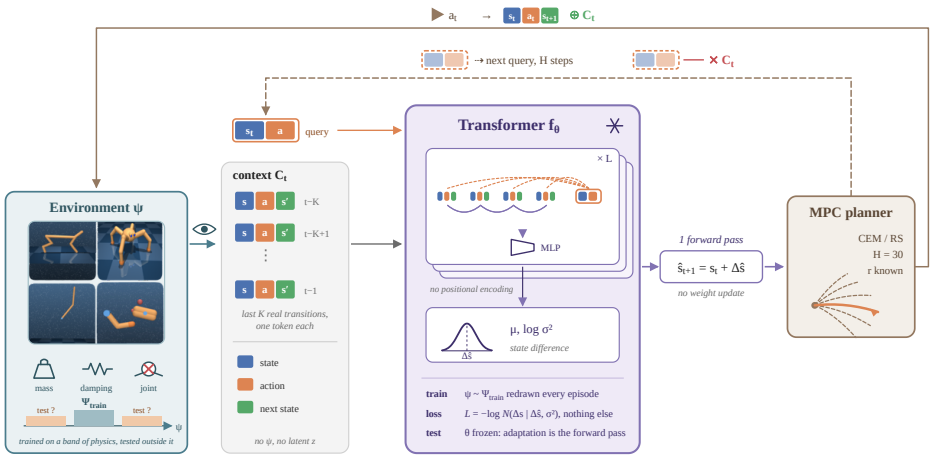


Figure 1: **Overview.** The last K real transitions of the episode form the context C_t, each (s, a, s') kept whole as one token (blue: state, orange: action, green: next state). A frozen transformer f_θ attends from the query (s_t, a) to these items and predicts $\hat{s}_{t+1}$ in one forward pass, no environment label, no latent code, no test-time gradient. An unchanged MPC planner rolls the model forward H steps with C_t held fixed (dashed), executes the first action, and appends the real transition to C_t (solid).

a key practical distinction from test-time fine-tuning and meta-learning approaches. In our experiments, we empirically quantify the value of this computational efficiency.

Ensembles. The ensemble baseline trains five independent models and propagates 20 particles over the planning horizon. Each particle is assigned to a specific ensemble member for the entire trajectory, following the trajectory sampling scheme of Chua et al. (2018). While this mechanism addresses epistemic uncertainty rather than adaptation, the two approaches are complementary, as demonstrated by our empirical results.

4.2 Training

To ensure a fair comparison, the training procedure matches the budget and iteration loop of Lee et al. (2020). Training spans 20 iterations; in each iteration, the agent collects 10 episodes using the current model and planner, then updates the network via gradient steps on windows of consecutive transitions sampled from the entire replay buffer. The training loss is the standard Gaussian negative log-likelihood of the state difference without auxiliary context losses, environment labels, or reconstruction terms. The model is trained once per environment family and evaluated directly on test environments without retraining, fine-tuning, or gradient-based adaptation. The total budget is 200k environment steps for MuJoCo tasks and 40k steps for classic control tasks, matching all model-based baselines in table 2. Each training window consists of a contiguous segment of transitions from a single episode, where the earlier portion serves as context, effectively capping the historical context length the model can learn to leverage. We conduct an empirical sweep over this window size, selecting the optimal length for each environment based solely on training return prior to test-time evaluation, adhering strictly to the benchmark's model-selection protocol.

4.3 Planning

We maintain the exact planning setup from Lee et al. (2020) to ensure that performance differences stem entirely from the dynamics model rather than the search procedure. For MuJoCo tasks, we use the Cross-Entropy Method (CEM) with 200 candidate action sequences, a planning horizon of 30 steps, 5 iterations, and 50 elites. For classic control tasks, we use Random Shooting with 1,000 candidate sequences. As in Lee et al. (2020) and PETS (Chua et al., 2018), the ground-truth reward function is assumed to be known. Candidate action sequences are evaluated by propagating trajectories forward from the current state using the dynamics model, keeping the historical context fixed throughout the rollout horizon. Unrolled (imagined) transitions are strictly excluded from the

Algorithm 1 Training and deployment (Nothing in the loop identifies the environment, and the weights are frozen once training ends)

Input: training parameter set Ψ_{train}, window W, context length K
initialise θ, buffer $\mathcal{D} \leftarrow \emptyset$
for iteration $= 1$ to 20 **do**
    for episode $= 1$ to 10 **do**
        draw $\psi \sim \Psi_{\text{train}}$ ▷ new physics each episode
        roll out with MPC under f_θ, appending transitions to $\mathcal{D}$
    end for
    for each gradient step **do**
        sample a window of W consecutive transitions from $\mathcal{D}$
        $\mathcal{L} \leftarrow$ Gaussian NLL of the state difference ▷ the only loss
        $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
    end for
end for

Deployment: θ is frozen. At step t in an unseen environment,
    $C_t \leftarrow$ the last K *real* transitions of this episode
    score candidate actions with $f_\theta(s_t, a \mid C_t)$, keeping C_t fixed while imagining
    execute the first action of the best sequence, observe s_{t+1}, append it to C_t

context. Because simulated states carry no ground-truth environment signal, appending them would cause the model to condition on its own predictions.

5 EXPERIMENTS

We ask four questions. Does in-context adaptation suffice for control under changing dynamics? Does the model read the context, or only benefit from its presence? What decides how much the context is worth? Where does the approach fail?

Protocol. We follow Lee et al. (2020) exactly. Environment parameters are drawn afresh at every episode from a training set, and testing uses parameters outside that set in two regimes, moderate and extreme. Training runs for 20 iterations of 10 episodes, which is 200k environment steps on MuJoCo and 40k on the classic-control tasks. Planning uses CEM on MuJoCo and random shooting elsewhere, with the reward function known. We report the model that reached the highest training return, which is the selection rule stated in that paper, and we average over five seeds. We report three arms of the model of section 4. SINGLE is one model, ENS is five models with 20 particles matching the PE-TS style of Chua et al. (2018), and NOCTX is the same architecture trained and evaluated without a context, which stands in for a plain dynamics model in our own simulator. Our numbers come from MuJoCo 3 and the published rows from mujoco-py 2.0; section G compares NOCTX against their vanilla dynamics model, the closest pair across the two simulators, and finds neither systematically ahead.

Dynamics generalization. As shown in table 2, our single model outperforms all non-ensemble baselines from Lee et al. (2020), including vanilla, stacked, and meta-learned dynamics models (GrBAL, ReBAL), across all six environments and evaluation regimes without requiring test-time gradient updates (e.g., 1760 vs. 1027 on HalfCheetah, -449 vs. -928 on Pendulum; see table 6). When augmented with an ensemble to match PE-TS, our approach surpasses PE-TS across all environments in the moderate range. Against the context-aware baseline (CaDM), our ensemble achieves superior performance on three of six environments by wide margins (e.g., 3277 vs. 2619 on CrippledHalfCheetah), while on Ant and SlimHumanoid it is ahead of both variants in both regimes as well, with the margin inside seed noise on Ant's moderate range (2143 against 2121 ± 60) and on SlimHumanoid's extreme one (1555 against 1487.9 ± 339). On HalfCheetah, where CaDM maintains an edge during extreme extrapolation, systematic ablations isolate the gap to out-of-distribution one-step prediction error rather than model capacity or search budget. Finally, while model-free baselines achieve higher returns on select tasks, they require up to $125\times$ more environment interactions (table 7).

Table 2: Dynamics generalization on the benchmark of Lee et al. (2020), average return over multiple seeds. Moderate and extreme are the two held-out parameter regimes. Published rows are reproduced from that paper and share our interaction budget; its model-free rows use 25 to 125 times that budget and are reported separately in table 7. Bold means best and underline the second-best.

Environment	Method	Moderate	Extreme
CartPole	**Ours, single model**	$\underline{187 \pm 4}$	$\underline{150 \pm 15}$
	Ours, ensemble	$\mathbf{196 \pm 1}$	$\mathbf{176 \pm 5}$
	Ours, no context	124 ± 4	112 ± 0
	Vanilla DM	124.7 ± 8.1	105.9 ± 3.7
	Stacked DM	131.9 ± 6.0	106.6 ± 11.0
	PE-TS	171.3 ± 18.5	142.9 ± 23.6
	Vanilla + CaDM	154.3 ± 13.2	118.6 ± 6.6
	PE-TS + CaDM	187.3 ± 11.2	149.4 ± 26.2
Pendulum	**Ours, single model**	$\underline{-449 \pm 17}$	$\underline{-767 \pm 60}$
	Ours, ensemble	$\mathbf{-434 \pm 11}$	$\mathbf{-681 \pm 133}$
	Ours, no context	-757 ± 45	-1119 ± 16
	Vanilla DM	-928.4 ± 56.8	-1170.6 ± 51.3
	Stacked DM	-1308.0 ± 49.6	-1288.3 ± 65.3
	PE-TS	-985.7 ± 64.2	-1221.1 ± 31.0
	Vanilla + CaDM	-593.7 ± 48.6	-967.9 ± 83.5
	PE-TS + CaDM	-705.5 ± 41.7	-1098.4 ± 47.1
HalfCheetah	**Ours, single model**	1760 ± 268	1136 ± 196
	Ours, ensemble	$\mathbf{3285 \pm 185}$	1007 ± 409
	Ours, no context	882 ± 230	301 ± 32
	Vanilla DM	1026.7 ± 164.7	686.7 ± 189.4
	Stacked DM	761.1 ± 236.6	661.5 ± 220.5
	PE-TS	2019.6 ± 274.8	$\underline{1422.3 \pm 162.8}$
	Vanilla + CaDM	2697 ± 385	1375 ± 118
	PE-TS + CaDM	$\underline{2934 \pm 782}$	$\mathbf{1627 \pm 506}$
CrippledHalfCheetah	**Ours, single model**	2082 ± 248	1273 ± 28
	Ours, ensemble	$\mathbf{3277 \pm 213}$	$\mathbf{1296 \pm 133}$
	Ours, no context	1170 ± 46	970 ± 46
	Vanilla DM	870.0 ± 308.0	577.3 ± 76.5
	Stacked DM	545.1 ± 289.8	417.9 ± 145.8
	PE-TS	1916.5 ± 328.2	1227.6 ± 35.2
	Vanilla + CaDM	1375.3 ± 290.6	966.9 ± 89.4
	PE-TS + CaDM	$\underline{2618.7 \pm 647.1}$	$\underline{1294.2 \pm 214.9}$
Ant	**Ours, single model**	1250 ± 93	778 ± 25
	Ours, ensemble	$\mathbf{2143 \pm 38}$	$\mathbf{1374 \pm 50}$
	Ours, no context	1001 ± 116	671 ± 50
	Vanilla DM	520.0 ± 97.6	385.8 ± 85.2
	Stacked DM	417.1 ± 46.8	338.9 ± 51.5
	PE-TS	1075.1 ± 103.6	856.6 ± 66.5
	Vanilla + CaDM	1315.7 ± 45.5	821.4 ± 113.5
	PE-TS + CaDM	$\underline{2121.0 \pm 60.4}$	$\underline{1200.7 \pm 21.8}$
SlimHumanoid	**Ours, single model**	$\underline{1596 \pm 224}$	$\mathbf{1566 \pm 291}$
	Ours, ensemble	$\mathbf{1619 \pm 237}$	$\underline{1555 \pm 140}$
	Ours, no context	1488 ± 161	1237 ± 106
	Vanilla DM	1004.4 ± 798.2	1155.5 ± 556.9
	Stacked DM	876.2 ± 1005.2	651.8 ± 449.9
	PE-TS	758.6 ± 528.8	810.4 ± 363.4
	Vanilla + CaDM	1228.9 ± 374.0	1487.9 ± 339.0
	PE-TS + CaDM	903.7 ± 343.9	814.5 ± 274.8

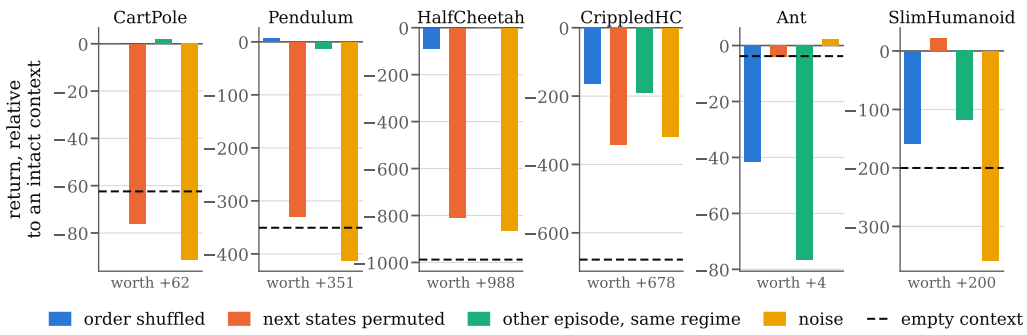


Figure 2: What each corruption of the context costs, in return, with the weights held fixed. The dashed line marks an empty context and the label below each panel gives what the context is worth there.

To verify that gains stem from genuine dynamics modeling rather than increased sequence length, we run two controls: test-time interventions that change what the context holds while the weights stay frozen, and retrained variants that change what a context item is.

Removing the context. We take a trained model, empty its context, and rerun the test suite, so nothing but the context differs. Figure 3 (left) reports all six environments with their individual seeds. HalfCheetah falls from 1760 to 656 on the moderate range and from 1136 to 214 on the extreme one, CrippledHalfCheetah by 651, Pendulum by 362, SlimHumanoid by 150 and CartPole by 65. Ant is the exception and loses nothing, which we return to in limitations. Table 8 gives the same numbers in a table.

Lengthening the context. Sweeping context length from zero to the full sequence window reveals rapid performance saturation (figure 4, table 12). On HalfCheetah, just two transitions nearly double the return from 656 to 1259, capturing half the overall adaptation gain, while four transitions recover 91% (1664 vs. 1760 at length 32). On Pendulum, performance saturates even faster, reaching -466 at length 8 compared to -465 at 32. That 2 to 4 transitions suffice indicates that the model rapidly identifies underlying physical parameters rather than merely memorizing trajectory history.

Corrupting the context. We evaluate four context corruptions while holding context length and shape fixed (section H and figure 2, table 9). Crucially, corruptions that destroy the action–outcome relationship, such as permuting next states (dropping HalfCheetah return from 1760 to 940) or replacing transitions with feature-matched Gaussian noise (dropping return to 830), degrade performance nearly to the no-context baseline (656). Conversely, shuffling transition order (1635) or borrowing context from a different episode in the same test regime (1691) causes minimal loss. This confirms that the model relies strictly on precise, paired input–output dynamics rather than unordered sequence sets or passive noise signatures.

Changing what the context is. Controlled retrainings confirm the necessity of complete, granular transition tuples (table 3). Dropping the next state from context items-leaving only state–action pairs-degrades HalfCheetah returns from 1760 to 1211 (moderate) and 1136 to 471 (extreme), as state–action pairs alone provide no signal for parameter identification. Averaging transition items across the context window incurs an even steeper penalty (dropping return to 1023), losing two-thirds of total context utility despite maintaining sequence length and history span. This demonstrates that the Transformer depends on fine-grained, individual transition dynamics rather than aggregated state statistics. The pooling row is the informative one, because the model is retrained under that constraint and so has every opportunity to learn to use a summary. What it learns is worth 1023 rather than 1760, so it addresses items individually rather than consuming a summary of them. Section J states what that does and does not establish about a learned context encoder.

A shift the model never trained on. Every environment in the benchmark varies mass, damping, or a disabled actuator, and gravity is fixed throughout. We rescale gravity at test time only, so the model faces an axis of variation that nothing in its training told it about (table 14). The context

Table 3: What the context has to be. Each row changes one thing about the context and retrains, against the single-model arm of table 2. Moderate / extreme test return, mean over seeds. Dropping the next state from an item removes what identifies the dynamics. Replacing the items by their average costs most of the benefit even though every transition is still present, so the model addresses items rather than consuming a summary of them. The order comparison in table 9 is a design check, since the model carries no positional encoding. What the training distribution has to be is a separate question and is reported in table 16.

Change	HalfCheetah	Pendulum	Ant
(reference) transitions, attended	1760/1136	−449/ − 767	1250/778
items are pairs, next state dropped	1211/471	−484/ − 1000	1258/850
items averaged into one token	1023/422	−453/ − 951	1235/815

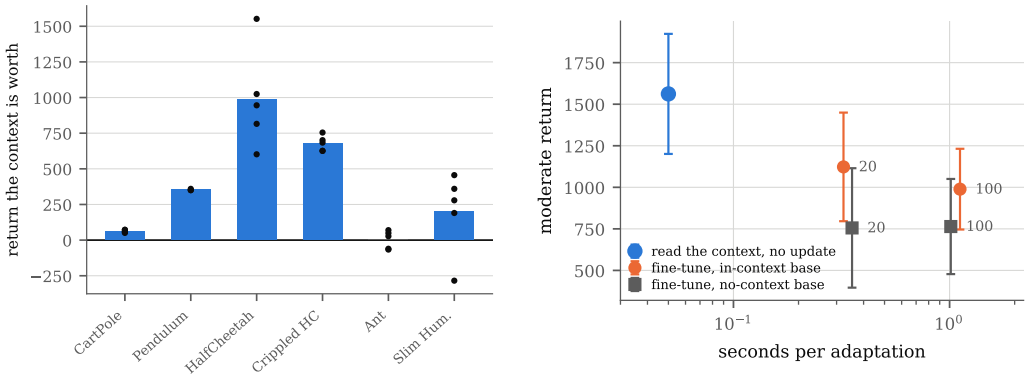


Figure 3: Left: what the context is worth in each environment, with the weights held fixed. Bars are the mean over five seeds and the dots are the five paired differences. Right: return against the cost of one adaptation on HalfCheetah, log scale.

keeps paying on every environment that read it under the benchmark's own shifts, by up to 682 return on HalfCheetah and by 110 to 277 across all four gravities on CrippledHalfCheetah, and is worth nothing on Ant here as everywhere else. What the model learned is therefore not tied to the axes it was trained on.

Value of context. Evaluated across individual held-out ranges, adaptation gains concentrate heavily on one side of the training distribution (table 18, figure 5). On HalfCheetah, the context already pays below the training parameters ($+241$, $t = 4.98$; $+289$, $t = 3.28$), and pays far more above them ($+1919$, $t = 5.22$; $+1603$, $t = 5.81$). Pendulum provides a direct control: across two equidistant, mirror-image shifts (one lighter/longer and one heavier/shorter), context gains jump from $+116$ on the former to $+607$ on the latter. This confirms that the direction of distribution shift, rather than its magnitude, governs context utility. SlimHumanoid exhibits the same directional asymmetry, whereas CartPole exhibits a ceiling effect due to high baseline returns without context (189 and 194 of a maximum near 200).

Impact of training. Nothing so far explains why the same architecture reads its context on one environment and ignores it on another. The training window does. It sets how many consecutive steps the model sees in one update, and therefore how much history it can learn to use. We sweep it with everything else held fixed, keeping the product of window and batch size constant so that every arm sees the same number of transitions per update. Table 4 reports the one step of that sweep which survives a paired test, and table 10 in section C the sweep in full. Going from a window of 8 to a window of 16 is worth 289 return on HalfCheetah, positive on all five seeds, and 240 in the extreme regime. Pendulum gains 75 in the extreme regime and nothing measurable in the moderate one. The same change on Ant is worth -9 and $+30$, neither separable from zero. We do not claim a dose-response law, since the step from 4 to 8 is not separable from zero on any environment and neither is the step from 4 to 16. What the sweep shows is that the training window moves in-context adaptation on two environments and does nothing on the third, with the architecture held fixed.

Table 4: Lengthening the training window from 8 to 16, everything else held fixed. The arms share seed numbers, so the difference is taken within a seed and the statistic is a paired t over the five of them. The full sweep over windows 2 to 16 is table 10.

Environment	Moderate		Extreme	
	difference	t	difference	t
Pendulum	$+36$	1.00	$+75$	$\mathbf{3.91}$
HalfCheetah	$+289$	$\mathbf{3.56}$	$+240$	$\mathbf{3.06}$
Ant	-9	-0.22	$+30$	0.82

Table 16 intervenes on the training distribution in both directions and finds the same environment-specific pattern.

The obvious alternative to reading a context is to fine-tune on it. We give both methods the same evidence, fine-tuning a copy of the trained weights on the last K real transitions before planning, where K is the context length of the in-context model (section I). Figure 3 (right) reports the comparison. Reading the context is both better and cheaper: fine-tuning for 20 or 100 steps loses about 40% of the return and costs a gradient computation per adaptation instead of one forward pass. The gap is not a matter of tuning the step count, since 100 steps is no better than 20, and it is not an artefact of a weak starting point, since fine-tuning the in-context model itself also makes it worse.

Prediction without the planner. Return confounds how well the model predicts with how well the planner searches, so we score prediction directly, on transitions collected with random actions in each test regime and with no planning involved (table 13). On Pendulum and CrippledHalfCheetah the context lowers one-step error, most cleanly on CrippledHalfCheetah where the spread across seeds is small. On HalfCheetah it leaves one-step error unchanged and only lowers it over ten steps. On Ant it lowers the error at neither horizon, which places Ant's failure in the model rather than in the planner: there is no accuracy for a planner to exploit.

Limitations. On HalfCheetah, PE-TS + CaDM stays ahead of our ensemble outside the training range, reaching 1627 against our 1007. Ablations rule out control capacity, model capacity, context length, training coverage, and planner budget; the remaining limitation is extrapolation to unseen dynamics. One-step error increases 80× on the light, weakly damped range, where ensemble averaging also provides little benefit because all models fail together. In-context adaptation can identify familiar dynamics but cannot compensate for unsupported extrapolation.

Ant provides a related limitation: its context produces essentially no adaptation, even after expanding training coverage, and we do not identify the cause. Its one-sided test range also makes it difficult to separate model and benchmark effects. More broadly, we study parametric dynamics shifts rather than changes to state/action spaces or rewards. SlimHumanoid provides limited evidence, and comparisons across MuJoCo 3 and mujoco-py 2.0 should be interpreted cautiously because simulator differences may contribute to performance gaps.

6 Conclusion

We asked whether in-context adaptation is enough for control when the physics change. On the six environments of Lee et al. (2020), under their protocol and budget, a plain sequence model over transitions beats the vanilla, stacked and meta-learned model-based baselines everywhere. With an ensemble it passes PE-TS on every environment and clears both context-aware CaDM variants outright on three of the six while staying inside seed noise on two more. The mechanism is measurable: removing the context, shortening it, breaking the pairing inside it, or replacing the transitions by their average each costs return, and each costs a different amount. What the model needs is whole transitions it can address one at a time. The recipe that follows is simple. Train once on a family of dynamics, keep whole transitions in the context, and adapt with forward passes. What it is worth depends on where a model is tested as much as on how it was trained, and Ant, the only environment that gains nothing, is also the only one tested on a single side of its training range, though that is not the whole of Ant. The open question is not how to build more adaptation machinery, but how to predict, before training, whether an environment yields a model that reads its context or one that memorizes the average.

Reproducibility statement

The training loop, losses, and planner settings are given in algorithm 1 and sections 4.2 and 4.3, and the per-environment settings in table 5. The supplemental material contains all the necessary scripts. The environment code is ported from the repository of Lee et al. (2020) with parameter sets, observations, rewards, and episode lengths unchanged (section G). Training used one GPU: a single-model run takes one to three hours, and an ensemble run takes six to eight hours on an A100, and test-time adaptation is a forward pass that requires no gradient computation.

AI use statement

We used generative AI tools to edit and proofread the manuscript for grammar, spelling, and consistency of terminology and notation, and to assist with reviewing the experimental code and formatting the bibliography. We did not use generative AI tools to propose research ideas, design the method or the experiments, formulate mathematical claims, or interpret results; the research questions, method, experiments, and analysis are our own. All AI-assisted edits and code reviews were verified by the authors, who take full responsibility for the final content of this work.

References

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A tutorial on meta-reinforcement learning. In *Foundations and Trends in Machine Learning: Vol. 18, No. 2-3, pp 224-384 (2025)*, 2023. doi: 10.1561/2200000080. URL http://arxiv.org/abs/2301.08028v4.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. Perk: Long-context reasoning as test-time learning. The Fourteenth International Conference on Learning Representations, 2026, 2025. URL http://arxiv.org/abs/2507.06415v3.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Frank Cole, Yulong Lu, Wuzhe Xu, and Tianhao Zhang. In-context learning of linear systems: Generalization theory and applications to operator learning. arXiv:2409.12293, 2024. URL http://arxiv.org/abs/2409.12293v3.

Jingtao Ding, Yunke Zhang, Yu Shang, Y. Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, Fengli Xu, and Yong Li. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025. doi: 10.1145/3746449. URL https://dl.acm.org/doi/pdf/10.1145/3746449.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. arXiv:1912.01603, 2019. URL http://arxiv.org/abs/1912.01603v3.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. arXiv:2310.16828, 2023. URL http://arxiv.org/abs/2310.16828v2.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, D. J. 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 (ICLR)*, 2023.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. In *International Journal of Computer Vision (2024)*, 2023. doi: 10.1007/s11263-024-02181-w. URL http://arxiv.org/abs/2303.15361v2.

Fan-Ming Luo, Tian Xu, Hang Lai, Xionghui Chen, Weinan Zhang, and Yang Yu. A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2), 2024. doi: 10.1007/s11432-022-3696-5.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023. doi: 10.1561/2200000086. URL https://www.nowpublishers.com/article/Download/MAL-086.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Anh Nguyen and Stefan Lee. Language-conditioned world model improves policy generalization by reading environmental descriptions. arXiv:2511.22904, 2025. URL http://arxiv.org/abs/2511.22904v1.

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

Sai Prasanna, Karim Farid, Raghu Rajan, and André Biedenkapp. Dreaming of many worlds: Learning contextual world models aids zero-shot generalization. arXiv:2403.10967, 2024. URL http://arxiv.org/abs/2403.10967v2.

Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. Worldgym: World model as an environment for policy evaluation. arXiv:2506.00613, 2025. URL http://arxiv.org/abs/2506.00613v3.

Rudolf Reiter, Jasper Justus Lutz Hoffmann, Dirk Peter Reinhardt, Florian Messerer, Katrin Baumgärtner, Shambhuraj Sawant, Joschka Bödecker, Moritz Diehl, and Sébastien Gros. Synthesis of model predictive control and reinforcement learning: Survey and classification. *Annual Reviews in Control*, 61:101045, 2026. doi: 10.1016/j.arcontrol.2026.101045. URL https://doi.org/10.1016/j.arcontrol.2026.101045.

Kevin Roice, Parham Mohammad Panahi, Scott M. Jordan, Adam White, and Martha White. A new view on planning in online reinforcement learning. arXiv:2406.01562, 2024. URL http://arxiv.org/abs/2406.01562v1.

Rylan Schaeffer, Mikail Khona, and Sanmi Koyejo. In-context learning of energy functions. arXiv:2406.12785, 2024. URL http://arxiv.org/abs/2406.12785v1.

R. Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. *arXiv (Cornell University)*, pp. 8583–8592, 2020. doi: 10.48550/arxiv.2005.05960. URL https://arxiv.org/pdf/2005.05960.

Richard S. Sutton, Csaba Szepesvari, Alborz Geramifard, and Michael P. Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. arXiv:1206.3285, 2012. URL http://arxiv.org/abs/1206.3285v1.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning (ICML)*, 2023.

Abdul Wahab, Raksha Kumaraswamy, and Martha White. Value bonuses using ensemble errors for exploration in reinforcement learning. Reinforcement Learning Journal, vol. 6, 2025, pp. 1894-1915, 2026. URL http://arxiv.org/abs/2602.12375v1.

Fan Wang, Zhiyuan Chen, Yuxuan Zhong, Sunjian Zheng, Pengtao Shao, Bo Yu, Shaoshan Liu, Jianan Wang, Ning Ding, Yang Cao, and Yu Kang. Context and diversity matter: The emergence of in-context learning in world models. 2026 International Conference on Learning Representations (ICLR), 2025. URL http://arxiv.org/abs/2509.22353v2.

Fan Wu, Jiacheng Wei, Ruibo Li, Yi Xu, Junyou Li, Deheng Ye, and Guosheng Lin. Ic-world: In-context generation for shared world modeling. arXiv:2512.02793, 2025. URL http://arxiv.org/abs/2512.02793v1.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems (RSS)*, 2017.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Yushuo Chen, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Peiyu Liu, Yiwen Hu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *Frontiers of Computer Science*, 20 (12), 2026. doi: 10.1007/s11704-026-60308-3.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *arXiv (Cornell University)*. Cornell University, 2024. doi: 10.48550/arxiv.2411.04983. URL https://arxiv.org/pdf/2411.04983.

A Implementation details

Model. The model is a transformer over transition tokens. Each context item is encoded from a state, an action and the resulting next state. The query token holds the current state and the candidate action. Attention runs from the query to the context items and to itself. The head predicts the mean and log-variance of the state difference, normalized by statistics of the collected data. The default size is four layers and width 128, which is a few million parameters.

Planning. The planner follows Lee et al. (2020), an instance of the broader combination of learned models with model-predictive control (Reiter et al., 2026). On MuJoCo it is the cross-entropy method with 200 candidates, horizon 30, five iterations, 50 elites and a truncated normal proposal. On classic-control tasks it is random shooting with 1000 candidates. During planning the context is encoded once per step and held fixed, so imagined states never enter the context.

Ensembles. The ensemble arm trains five models independently and propagates 20 particles, each assigned to one model for the whole horizon, in the style of trajectory sampling (Chua et al., 2018).

Per-environment settings. Two settings were tuned per environment, the number of gradient updates per iteration and the training window. Both were chosen by training return on seeds 0 and 1, before any test number was inspected, which mirrors the model-selection rule of the benchmark. Table 5 lists the chosen values.

Table 5: Per-environment settings, chosen by training return on seeds 0 and 1 before any test number was inspected. Updates is the number of gradient steps per iteration. Window is the length of the training window, and context is the number of transitions the model reads at decision time.

Environment	Updates	Window	Context
CartPole	2000	64	32
Pendulum	5000	64	32
HalfCheetah	2000	64	32
CrippledHalfCheetah	5000	64	32
Ant	5000	4	4
SlimHumanoid	2000	64	32

B Discussion

The context carries dynamics information, not just tokens. Three interventions point the same way. Emptying the context costs most of the gain. Lengthening it recovers the gain, and two transitions already recover half of it. Breaking the pairing between a state-action and its next state is almost as bad as having no context at all, and so is replacing the transitions by noise with the same mean and variance. A model that merely used the extra tokens as computation would not show that pattern, since the corrupted contexts are the same length and the same shape as the intact one.

One of the four interventions turns out to test something weaker than it appears, and we report it as such. Swapping contexts between the episodes that run in parallel costs little and on HalfCheetah costs nothing. Those episodes draw their parameters from the same held-out range, so the swapped context is a genuine history recorded under nearly the same physics, and the result says only that the model does not need its own episode's history in particular. It does not test what a context from genuinely different physics would do. What the corruption study does separate is the pairing inside a transition and its numerical content: break either and the context becomes almost worthless.

Items, not a summary. The model has no positional encoding, so the fact that shuffling the context costs little says that the architecture behaves as designed. The pooling result is what constrains the mechanism. Averaging the same transitions costs most of the benefit, so the model addresses items individually rather than consuming a summary of them. This is the shape of a regression or nearest-neighbour computation over the items, not of a learned latent code, and it explains why a context encoder is not the only route to adaptation. An encoder commits to its summary before the model has seen the query.

The training regime decides whether the context is read at all. The architecture is the same everywhere, so whatever decides this is not in the architecture. The training window decides part of it: lengthening it from 8 to 16 is worth 288 return on HalfCheetah on every one of five seeds and is worth nothing on Ant. The training distribution decides part of it on one environment, where the value of the context grows from 680 to 1040 as the parameter set is widened, and decides nothing on the other two we tested.

We are careful not to turn this into a law. The tempting one, that a model reads its context only when training makes reading necessary, predicts that Ant would start reading once its training set is widened, and it does not (table 16). What we can say is weaker and still useful: in-context adaptation is not obtained by adding a module, it is obtained or lost through the training regime, it varies by environment, and on Ant we cannot yet obtain it at all. Ant also shows that the two things come apart. Trained with a context it reaches 1182 where the same architecture trained without one reaches 921, yet emptying that context at test time costs it nothing: training with a context produced better weights rather than a model that reads.

Ensembles and context are complementary, up to a point. The ensemble improves the moderate range on all six environments, and on top of the context rather than instead of it, since the no-context ensemble stays below the in-context one. The exception marks where that stops: on HalfCheetah it is worth $+1133$ on the moderate range and -55 on the extreme one, buying nothing exactly where the single model's predictions fail. Averaging over disagreeing models protects a planner only while some of the models are right.

C ADDITIONAL RESULTS

Table 6: The meta-learned adaptive baselines of Lee et al. (2020), GrBAL and ReBAL, reproduced from that paper. They share our interaction budget and belong with table 2; they are shown here only because they are below every other method on every environment, often by an order of magnitude, and carrying them through the main table costs twelve rows.

Environment	Method	Moderate	Extreme
CartPole	GrBAL	87.5 ± 35.1	81.6 ± 21.3
CartPole	ReBAL	48.4 ± 34.5	54.3 ± 37.0
Pendulum	GrBAL	-1027.7 ± 68.1	-1048.1 ± 111.8
Pendulum	ReBAL	-860.8 ± 140.3	-1026.4 ± 51.5
HalfCheetah	GrBAL	-43.7 ± 106.9	-94.5 ± 141.3
HalfCheetah	ReBAL	657.5 ± 184.9	396.6 ± 188.5
CrippledHalfCheetah	GrBAL	-9.2 ± 17.1	16.6 ± 23.0
CrippledHalfCheetah	ReBAL	904.5 ± 90.7	833.0 ± 118.0
Ant	GrBAL	46.5 ± 6.5	42.9 ± 3.8
Ant	ReBAL	73.1 ± 15.5	53.0 ± 17.2
SlimHumanoid	GrBAL	-562.8 ± 253.5	-398.6 ± 177.2
SlimHumanoid	ReBAL	85.8 ± 388.9	108.7 ± 357.6

Table 12 gives the numbers behind figure 4, the context-length sweep with the weights held fixed, and table 13 the prediction errors behind the mechanism. The full per-run record, including every seed, configuration and the machine each run used, is kept with the code.

Table 14 reports the gravity shift in full, and table 15 an evaluation not discussed in the main text: the environment's parameters change silently half way through an episode, so a model cannot be retrained and whatever recovery happens has to come from the context it keeps reading. On HalfCheetah's hardest range the per-step return after the change is 0.34 with the context, 0.12 with the same weights and an emptied context, and 0.16 for a model trained without one. On Ant the no-context model is the better of the three, as everywhere else. CartPole is absent because the evaluation is defined only for fixed-length episodes.

Table 16 intervenes on the training distribution in both directions. Thinning it alone would only show that taking something away costs something, so the table also reports arms trained on an

Table 7: The model-free rows of Lee et al. (2020), reproduced from that paper. They are trained with 25 to 125 times the interaction budget of every method in table 2, so they are not a like-for-like comparison. With that budget they beat every model-based method, ours included, on CartPole, Pendulum and SlimHumanoid.

Environment	Method	Moderate	Extreme
CartPole	Vanilla PPO	199.1 ± 0.9	187.8 ± 4.7
	Stacked PPO	197.8 ± 1.3	189.2 ± 6.1
	PPO + PC	198.0 ± 1.4	187.5 ± 10.9
	PPO + EP	196.3 ± 4.0	184.5 ± 9.7
	PPO + CaDM	197.9 ± 3.0	193.0 ± 3.5
Pendulum	Vanilla PPO	-1113.2 ± 69.1	-1356.8 ± 48.0
	Stacked PPO	-475.7 ± 228.1	-488.2 ± 178.2
	PPO + PC	-645.3 ± 320.7	-1136.6 ± 251.0
	PPO + EP	-374.3 ± 24.6	-256.7 ± 26.4
	PPO + CaDM	-279.8 ± 42.1	-426.4 ± 227.0
HalfCheetah	Vanilla PPO	807.7 ± 553.6	574.0 ± 645.6
	Stacked PPO	361.1 ± 141.7	5.7 ± 208.1
	PPO + PC	642.1 ± 488.3	462.1 ± 534.5
	PPO + EP	895.3 ± 445.1	674.2 ± 686.8
	PPO + CaDM	1224.2 ± 630.0	1021.1 ± 676.6
CrippledHalfCheetah	Vanilla PPO	1223.6 ± 559.9	781.7 ± 270.3
	Stacked PPO	967.1 ± 146.6	904.4 ± 146.5
	PPO + PC	1162.2 ± 456.5	546.3 ± 215.9
	PPO + EP	1017.0 ± 201.1	719.0 ± 438.5
	PPO + CaDM	1454.0 ± 462.6	1025.0 ± 296.2
Ant	Vanilla PPO	149.4 ± 27.0	117.3 ± 23.1
	Stacked PPO	53.2 ± 10.6	46.0 ± 10.9
	PPO + PC	207.0 ± 33.8	163.5 ± 30.4
	PPO + EP	107.8 ± 19.9	93.5 ± 32.4
	PPO + CaDM	228.8 ± 48.4	199.2 ± 52.1
SlimHumanoid	Vanilla PPO	3761.3 ± 1582.4	2751.6 ± 869.4
	Stacked PPO	2443.0 ± 535.6	1577.8 ± 573.5
	PPO + PC	3928.5 ± 1848.7	2362.6 ± 781.9
	PPO + EP	2224.7 ± 882.9	1293.4 ± 729.0
	PPO + CaDM	4975.7 ± 1305.7	3015.1 ± 1508.3

Table 8: The effect of the context alone. The weights are those of the single-model arm of table 2; the interventions use the single model throughout because it has one causal path, whereas with an ensemble a change to the context also changes how the five members disagree, which is what the planner scores. Only the context given at decision time changes. Moderate / extreme test return, mean over five seeds. Context helps in five environments and is worth nothing in Ant.

Environment	Context as trained	Context emptied	Difference
CartPole	$187/150$	$123/111$	$+65/+40$
Pendulum	$-458/-788$	$-817/-1104$	$+358/+316$
HalfCheetah	$1760/1136$	$656/214$	$+1104/+922$
CrippledHalfCheetah	$2082/1273$	$1431/1103$	$+651/+170$
Ant	$1182/760$	$1188/805$	$-6/-45$
SlimHumanoid	$1596/1566$	$1446/1375$	$+150/+190$

Table 9: Corrupting the context at test time, with the weights held fixed; the single-model arm. Moderate / extreme test return. The intact column is the single-model arm of table 2, and the emptied comparison is in table 8.

Environment	intact	order shuffled	pairing broken	other episode	noise
CartPole	$187/150$	$180/145$	$130/113$	$188/132$	$105/95$
Pendulum	$-449/-767$	$-461/-823$	$-847/-1066$	$-483/-855$	$-932/-1102$
HalfCheetah	$1760/1136$	$1635/881$	$940/373$	$1691/905$	$830/314$
CrippledHalfCheetah	$2082/1273$	$1772/1247$	$1545/1130$	$1646/1249$	$1546/1163$
Ant	$1250/778$	$1210/828$	$1245/808$	$1171/780$	$1255/803$
SlimHumanoid	$1596/1566$	$1433/1472$	$1687/1681$	$1466/1524$	$1173/1495$

Table 10: Shortening the training window. Everything else is unchanged. Moderate / extreme test return, mean over seeds. The third column evaluates the same weights with the context emptied, and the fourth column is the return the context is worth. The chosen windows are 64 on HalfCheetah and Pendulum and 4 on Ant (table 5).

Environment	Window	as trained	context emptied	value of context
Pendulum	4	-525 ± 21 / -1019 ± 28	-884 ± 10 / -1074 ± 24	359 ± 27 / 55 ± 42
Pendulum	8	-477 ± 18 / -966 ± 74	-807 ± 31 / -1110 ± 32	330 ± 42 / 144 ± 81
Pendulum	16	-458 ± 17 / -855 ± 46	-856 ± 31 / -1100 ± 28	398 ± 15 / 245 ± 74
HalfCheetah	4	1392 ± 339 / 562 ± 205	986 ± 252 / 406 ± 162	407 ± 90 / 156 ± 117
HalfCheetah	8	1054 ± 57 / 425 ± 92	767 ± 123 / 320 ± 93	287 ± 154 / 104 ± 179
HalfCheetah	16	1295 ± 229 / 595 ± 198	555 ± 107 / 69 ± 62	741 ± 135 / 526 ± 203
Ant	2	1267 ± 142 / 793 ± 100	1321 ± 155 / 838 ± 104	-54 ± 29 / -44 ± 14
Ant	4	1250 ± 93 / 778 ± 25	1238 ± 41 / 857 ± 86	12 ± 54 / -79 ± 88
Ant	8	935 ± 81 / 604 ± 88	911 ± 125 / 643 ± 80	24 ± 45 / -39 ± 49
Ant	16	748 ± 77 / 522 ± 75	725 ± 82 / 533 ± 51	23 ± 49 / -11 ± 31

Table 11: Test-time gradient adaptation, the alternative to reading. Every adaptation reads the same $k = 32$ most recent transitions the in-context model would read, takes 20 or 100 gradient steps at learning rate 10^{-4}, and starts again from its original checkpoint next time, so the two arms see the same evidence and differ only in how they use it. Two starting points: the in-context model, which also keeps reading its context while adapting, and the no-context model of table 2. The in-context (reference) row is the single-model arm of table 2, which takes no gradient step. Moderate / extreme test return, mean over seeds; the last column is the measured wall-clock cost of one adaptation on a shared GPU.

Environment	Adaptation	Moderate	Extreme	s / adaptation
HalfCheetah	in-context (reference)	1760 ± 268	1136 ± 196	< 0.1
HalfCheetah	fine-tune, in-context base, 20 steps	1098 ± 76	647 ± 208	0.4
HalfCheetah	fine-tune, in-context base, 100 steps	987 ± 103	636 ± 65	1.1
HalfCheetah	fine-tune, no-context base, 20 steps	884 ± 315	339 ± 144	0.2
HalfCheetah	fine-tune, no-context base, 100 steps	832 ± 254	404 ± 178	0.9
Pendulum	in-context (reference)	-449 ± 17	-767 ± 60	< 0.1
Pendulum	fine-tune, in-context base, 20 steps	-519 ± 7	-690 ± 117	1.5
Pendulum	fine-tune, in-context base, 100 steps	-513 ± 17	-648 ± 56	1.3
Pendulum	fine-tune, no-context base, 20 steps	-668 ± 8	-999 ± 33	0.7
Pendulum	fine-tune, no-context base, 100 steps	-556 ± 25	-778 ± 55	1.1

enriched parameter set. The three environments behave differently. On HalfCheetah the value of the context grows with the training set, from $+979$ to $+1288$. On Ant it stays at zero across a five-fold widening, which rules out low training diversity as the reason Ant does not read its context. On Pendulum removing almost all of the diversity costs nothing at all. We therefore do not claim that training diversity is what produces in-context adaptation. It matters on one environment of the three, and even there the thinned set keeps most of the value, $+979$ of $+1104$.

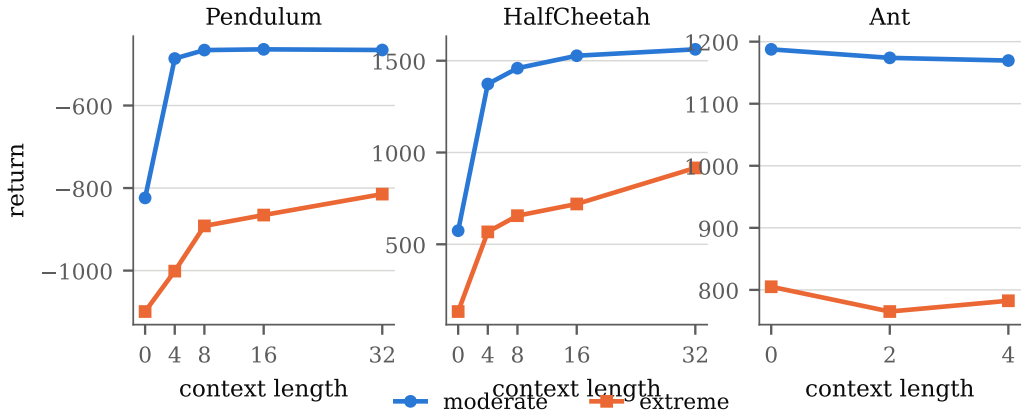


Figure 4: Return against the number of transitions the model reads, with the weights held fixed. Length zero empties the context of a model that was trained with one. Performance saturates within a few transitions on the two environments that read their context, and Ant is flat.

Table 12: Context length at test time, with the weights held fixed. The weights are those of the single-model arm of table 2; the interventions use the single model throughout because it has one causal path, whereas with an ensemble a change to the context also changes how the five members disagree, which is what the planner scores. Length zero empties the context of a model trained with one. Moderate / extreme test return, mean over seeds; the Ant sweep covers the first three lengths.

Context length	Pendulum	HalfCheetah	Ant
$T = 0$	$-824/-1099$	$656/214$	$1188/805$
$T = 2$	$-544/-1048$	$1259/551$	$1174/765$
$T = 4$	$-486/-1001$	$1664/752$	$1182/760$
$T = 8$	$-466/-892$	$1726/820$	–
$T = 16$	$-464/-865$	$1704/823$	–
$T = 32$	$-465/-815$	$1760/1136$	–
$T = 64$	$-449/-855$	$1824/983$	–

Table 13: Prediction on the held-out parameter ranges, with the weights held fixed: context as trained against an emptied context, on random-action rollouts, mean squared error averaged over the held-out ranges. CartPole was not part of this evaluation.

Environment	One step, context	One step, no context	Ten steps, context	Ten steps, no context
Pendulum	0.0360 ± 0.0029	0.0532 ± 0.0076	0.507 ± 0.041	0.820 ± 0.109
HalfCheetah	0.3930 ± 0.0453	0.3985 ± 0.0417	0.885 ± 0.077	0.954 ± 0.060
CrippledHalfCheetah	0.0902 ± 0.0030	0.1053 ± 0.0033	0.272 ± 0.009	0.314 ± 0.012
Ant	0.0351 ± 0.0029	0.0341 ± 0.0030	0.214 ± 0.018	0.209 ± 0.018
SlimHumanoid	0.1701 ± 0.0070	0.1727 ± 0.0064	0.461 ± 0.021	0.485 ± 0.023

D WHERE THE CONTEXT PAYS

Table 17 sets out how the benchmark of Lee et al. (2020) varies each environment. Ant is the only one with a single varying parameter, since its damping is pinned at 1.0 in training and in both held-out ranges, and the only one whose held-out values all sit on one side of the training range. This is a property of the published parameter sets and can be read off them directly.

Table 18 splits the value of the context by held-out range instead of averaging it into one number per environment. On HalfCheetah, Pendulum and SlimHumanoid the benefit sits above the training range. CartPole is the apparent exception, and the resolution is a ceiling: with a strong actuator it

Table 14: A shift the model never trained on. Gravity is fixed in every training and test environment of the benchmark, and is rescaled here at test time only. Same weights (the single-model arm), context as trained against an emptied context, averaged over the test ranges and over seeds.

Environment	$0.5\times$	$0.75\times$	$1.25\times$	$1.5\times$
CartPole	$+20$	$+54$	$+38$	$+16$
Pendulum	$+329$	$+265$	$+59$	$+13$
HalfCheetah	-10	$+682$	$+436$	$+277$
CrippledHalfCheetah	$+277$	$+194$	$+174$	$+110$
Ant	$+9$	-44	-36	-15
SlimHumanoid	$+224$	$+433$	$+267$	-14

Table 15: A silent mid-episode switch. Each episode starts with training-distribution parameters; at half the episode the parameters are redrawn from a test range while the state is kept, and no gradient is taken. Each cell is the mean per-step reward over the first 50 steps after the switch over the mean over the remaining steps, averaged over seeds. Context as trained reads 32 transitions, emptied reads none, and NoCtx is the model trained and evaluated without context. All rows are the single-model arm.

Environment	Range	as trained	emptied	NoCtx
Pendulum	0 (extreme)	$-0.01/-0.01$	$-0.09/-0.01$	$-0.49/-0.25$
Pendulum	1 (moderate)	$-0.15/0.00$	$-0.26/0.00$	$-0.67/-0.30$
Pendulum	2 (moderate)	$-0.02/-0.02$	$-0.19/-0.11$	$-0.17/-0.01$
Pendulum	3 (extreme)	$-4.17/-5.17$	$-4.34/-5.34$	$-4.98/-5.75$
HalfCheetah	0 (extreme)	$2.62/0.34$	$1.57/0.12$	$2.39/0.16$
HalfCheetah	1 (moderate)	$2.75/0.89$	$1.70/0.49$	$2.22/0.73$
HalfCheetah	2 (moderate)	$1.72/1.15$	$1.12/0.37$	$1.73/0.84$
HalfCheetah	3 (extreme)	$1.71/1.14$	$0.62/0.11$	$1.23/0.33$
CrippledHalfCheetah	0 (moderate)	$1.79/1.80$	–	$0.93/0.88$
CrippledHalfCheetah	1 (extreme)	$1.70/1.22$	–	$1.00/0.75$
Ant	0 (extreme)	$0.63/0.47$	$0.53/0.48$	$0.60/0.69$
Ant	1 (moderate)	$0.59/0.59$	$0.68/0.60$	$1.39/0.85$
SlimHumanoid	0 (extreme)	$0.87/0.58$	–	$-0.26/1.05$
SlimHumanoid	1 (moderate)	$-0.17/0.22$	–	$0.63/0.41$
SlimHumanoid	2 (moderate)	$0.18/0.10$	–	$0.64/0.18$
SlimHumanoid	3 (extreme)	$0.31/0.09$	–	$-0.37/-0.10$

already scores 191 to 194 with the context emptied against a maximum near 200, so there is nothing for a context to add. Ant is measured in the below-training direction only, which is where the other environments also show the least.

Table 19 tests that reading directly. It takes HalfCheetah's trained weights, freezes them, keeps the planner unchanged, and evaluates on four constructed ranges that cross two factors, whether the band of mass sits below or above the training range, and whether damping varies with it or stays at a trained value. Damping 1.0 is inside HalfCheetah's training set, so the single-parameter rows ask nothing extra of the model. Only the direction matters. Figure 5 plots the two tables side by side.

E Compute

Every run uses a single GPU. A single-model run takes one to three hours and an ensemble run takes six to eight hours on an A100. Adaptation at test time costs one forward pass per planning step and no gradient computation, which is the practical difference from test-time fine-tuning.

F HalfCheetah, in detail

The gap to PE-TS with CaDM on HalfCheetah's extreme range is the largest single result we cannot explain away, so we record what we ruled out. Training return places the failure in extrapolation rather than in control: our ensemble reaches 6132 on the training parameters against 8264 for that method, a factor of 1.3, while on the held-out ranges the factor is 2.6 and then 5.4. Capacity is not the cause. A six-layer model reaches 1119 on the moderate range and a width-256 model 841,

Table 16: How much the training distribution matters. The training parameter set is thinned and enriched, everything else held fixed, and the quantity reported is what the context is then worth at test time: the return with the context as trained minus the return of the same weights with the context emptied, computed per seed on the moderate ranges. Mean $\pm$ standard deviation over seeds. On HalfCheetah the value of the context grows with the training set. On Ant it stays at zero however wide the training set is made, which rules out low training diversity as the reason Ant does not read its context. On Pendulum removing almost all of the diversity costs nothing.

Environment	Training set	Value of the context
HalfCheetah	thinned to 2×2	$+979 \pm 288$
	as reported, 5×5	$+1104 \pm 321$
	enriched down to 0.4	$+1288 \pm 97$
Ant	thinned to 2×1	-27 ± 73
	as reported, 4×1	-16 ± 44
	enriched to 7×3	-30 ± 28
	enriched to 4×4	-19 ± 147
Pendulum	thinned to 2×2	$+408 \pm 12$
	as reported, 11×11	$+358 \pm 15$

Table 17: How the benchmark of Lee et al. (2020) varies each environment, against what the context is worth there. Ant is the only environment with a single varying parameter, and the only one whose held-out values all lie on one side of the training range. The first parameter's ranges are shown; CrippledHalfCheetah varies which actuators are disabled rather than a continuous quantity. The last column is the value of the context, moderate ranges, paired within a seed (table 8).

Environment	Varying	Parameters	Train	Held out	Both sides	Context
HalfCheetah	2	mass, damping	0.75–1.25	0.2–1.8	yes	$+1104$
CrippledHalfCheetah	2	joint, count	0–3	4–5	no	$+651$
Pendulum	2	mass, length	0.75–1.25	0.2–1.8	yes	$+362$
SlimHumanoid	2	mass, damping	0.8–1.25	0.4–1.8	yes	$+150$
CartPole	2	force, length	5–15	2–18	yes	$+65$
Ant	1	mass	0.85–1	0.2–0.6	no	$+12$

against 1562 for the same four-layer, width-128 model over the same five seeds, and the two changes together reach 693. Every one of them trains better and tests worse. A longer test-time context does not help either, at 1207 for length 64. Widening the training set downwards to cover the light regime raises what the context is worth, from $+988$ to $+1040$, but lowers the absolute return to 1225, since the same capacity now has to cover more ground.

Search is not the limit. Re-evaluating the trained weights with two and a half times the planner budget, 500 CEM candidates over 8 iterations with 100 elites instead of 200 over 5 with 50, changes the moderate return by $+106$ on average over five seeds. The individual seeds are -26, $+16$, $+371$, $+170$ and -2, a spread that overlaps zero and is far smaller than the seed-to-seed spread of the arm itself, which is ± 361.

What is left is prediction. One-step error on the light, weakly damped range is eighty times the error on the training parameters, and the ensemble, which is worth $+1525$ on the moderate range, is worth -129 on the extreme one. Averaging over five models protects a planner while some of them are right, and stops protecting it when they fail together.

G THE SIMULATOR PORT

Our numbers come from MuJoCo 3 and the published rows from mujoco-py 2.0, so a reader should ask whether the simulator explains any gap. Our no-context arm is the closest thing we have to their vanilla dynamics model, since both train one model on the same data with no context mechanism, and across the six environments the two land in the same range with neither systematically ahead. Ours is higher on Ant, CrippledHalfCheetah, SlimHumanoid and Pendulum, lower on HalfCheetah, and close on CartPole. Differences of that shape are what the port costs, and they are smaller than

Table 18: What the context is worth, split by held-out range, rather than averaged into one number per environment. *Side* is whether the range's first parameter falls below or above the training range. On HalfCheetah, Pendulum and SlimHumanoid the value is concentrated above; on CartPole the ranges worth nothing are the ones already solved without a context. Ant is tested below only. Paired within a seed, with t over the pairs.

Environment	Range	Side	Context	Emptied	Value	t
HalfCheetah	0.2–0.3	below	376	135	+241	+4.98
	0.4–0.5	below	843	554	+289	+3.28
	1.5–1.6	above	2677	758	+1919	+5.22
	1.7–1.8	above	1895	292	+1603	+5.81
	0.2–0.4	below	−304	−536	+232	+3.28
	0.5–0.7	below	−423	−540	+116	+11.82
	1.3–1.5	above	−475	−1083	+607	+18.78
	1.6–1.8	above	−1230	−1681	+452	+4.90
	0.4–0.5	below	1693	1601	+93	+1.15
	0.6–0.7	below	1735	1871	−136	−0.30
	1.5–1.6	above	1457	1020	+437	+5.32
	1.7–1.8	above	1438	1150	+288	+2.22
	2–2.5	below	107	30	+77	+4.11
	3–3.5	below	186	51	+135	+16.87
	16.5–17	above	189	194	−6	−0.88
	17.5–18	above	194	191	+3	+0.30
	4–5	above	2082	1431	+651	+26.06
	4–5	above	1273	1103	+170	+1.78
	0.2–0.4	below	778	857	−79	−1.26
	0.4–0.6	below	1250	1238	+12	+0.32

Table 19: The same HalfCheetah weights, frozen, on four constructed held-out ranges crossing two factors, whether the band of mass sits below or above the training range and whether damping varies with it or stays at a trained value. Only the direction matters. The first row is the shape the benchmark gives Ant. Mean over seeds, paired within a seed.

Held-out range	Context	Emptied	Value
light, one axis (Ant-shaped)	1152	779	$+373 \pm 122$
light, two axes	1018	561	$+458 \pm 68$
heavy, one axis	3377	1529	$+1848 \pm 642$
heavy, two axes	2243	524	$+1719 \pm 635$

the effects we report. We ported the environment code from the original repository and kept the parameter sets, observations, rewards and episode lengths unchanged.

H THE FOUR CORRUPTIONS

Each corruption acts on the context tensor after it is assembled and before the model reads it, with the weights frozen.

Order shuffled permutes the transitions in time, moving the three parts of each item together so that every item stays intact. It is a check rather than a discovery: the model carries no positional encoding, so its cost has to be small, and we report it to confirm that the implementation behaves as the architecture says it should.

Next states permuted permutes only the next states, so each item remains a well-formed triple of the right shape and scale while the pairing between a state-action and its outcome is destroyed. This is the intervention that isolates what identifies the dynamics.

Other episode, same regime rolls the context along the batch of episodes that run in parallel. Those episodes draw their parameters from the same held-out range, so the swapped context is a real history

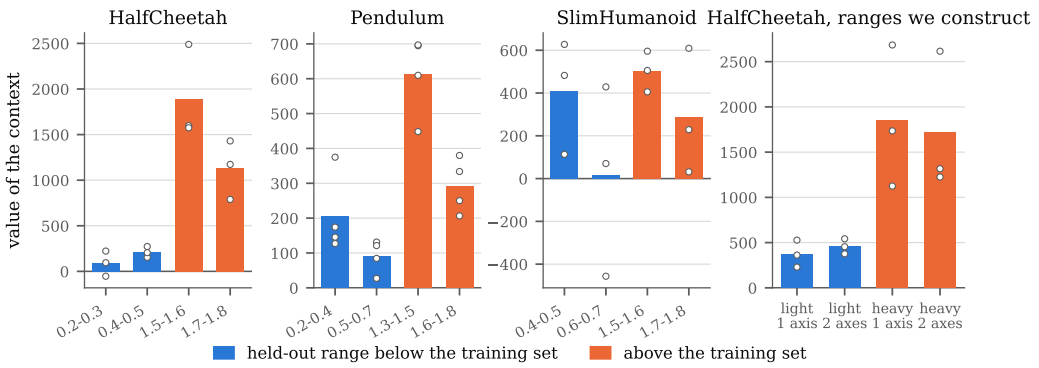


Figure 5: What the context is worth, split by held-out range. Left three panels: the environments for which the below/above contrast is informative. Each bar is one range, its height is the paired difference of the means, the dots are the per-seed paired differences, and the colour marks which side of the training set the range sits on. Right panel: the same frozen HalfCheetah weights on four constructed ranges crossing the direction of the mass shift with whether damping varies alongside it.

recorded under nearly the same physics. The intervention therefore asks whether a model needs its own episode's history in particular, and not what a context from different physics would do.

Noise replaces every entry with samples that match the mean and standard deviation of the intact context, removing the information while keeping the shape and scale.

Two of the four are informative and two are not. Permuting the next states and replacing them with noise each cost almost everything, which is the result reported in the main text. The other two cost almost nothing, and for different reasons. Shuffling the order has to cost little, because the model carries no positional encoding and its attention over the context is therefore permutation invariant by construction; we report it as a check that the implementation behaves as the architecture says, not as a finding. Swapping contexts between parallel episodes is weaker than it looks, because those episodes draw their parameters from the same held-out range: the swapped context is a genuine history recorded under nearly the same physics. It shows that a model does not need its own episode's history in particular. It does not show what a context recorded under different physics would do, which would be the sharper test and which we did not manage to run correctly.

I TEST-TIME FINE-TUNING

The procedure follows the standard recipe of test-time adaptation (Liang et al., 2023). During each test episode, we take the last K real transitions, where K is the context length of the in-context model, and fine-tune a copy of the trained weights on them before planning, so that both methods see exactly the same evidence and differ only in how they use it. Adaptation restarts from the trained weights every time rather than accumulating, which mirrors a sliding context window: letting it accumulate would give the gradient method the whole episode while the in-context model still sees only K transitions. Re-adapting at every step would dominate the run, so it happens every few steps and we record the wall-clock cost of one adaptation.

We report two starting points. One is the no-context model, which is what a gradient method would naturally start from. The other is the in-context model itself, which keeps reading its context while being fine-tuned; without that it would be penalized twice. Table 11 keeps the two apart. The in-context model reaches 1098 after 20 steps against 1760 with no update at all, so adding gradient steps to a model that already reads its context makes it worse. A plausible reading is that a handful of gradient steps on 32 transitions, taken in a parameter regime the model never saw, is as likely to damage the weights as to improve them, while reading a context cannot damage anything.

J WHAT THE POOLING RESULT ESTABLISHES

Replacing the context items by their running average costs HalfCheetah 1760 to 1023, two-thirds of what the context is worth there. The comparison is tight in one respect and loose in another, and both matter for how the result should be read.

It is tight because nothing else changes. The encoder, the parameter count, the training data and the number of updates are identical, and the averaged context covers exactly the same span of history as the addressable one. The model is retrained under the constraint rather than having it imposed at test time, so it has every opportunity to learn to make the best of a summary. What it learns is worth 1023. The comparison is also not a restatement of the order-shuffling result: the model carries no positional encoding, so permutation invariance is architectural, whereas nothing in the architecture makes a mean as useful as the items it averages.

It is loose because we are measuring our own architecture under a bottleneck, not the bottleneck a context encoder learns. A summary trained with its own objective, as CaDM's is, may well carry more than an unweighted mean does, and we do not claim otherwise. What the row establishes is narrower and still useful: the difference between addressing transitions and summarizing them is not a matter of implementation. On this environment, it is worth two-thirds of the context's value, which places a cost on any design that commits to a summary before it has seen the query.