Learning When to Think: Adaptive Internal Computation for Reinforcement Learning

Rohit Kumar Salla **Simon Stepputtis**
Virginia Polytechnic Institute and State University
{rohits25, stepputtis}@vt.edu

Abstract

Iterative neural policies usually spend the same computation on every decision, although the benefit of additional processing varies across states. We introduce Adaptive Internal Computation (AIC), an iterative policy that learns when to stop refining its latent representation and act. In AIC, we train action selection and stopping jointly, an expected-depth penalty values computation and intermediate value predictions provide auxiliary supervision. We show that when only some states benefit from further refinement, an ideal state-dependent allocator outperforms every state-independent stopping schedule at the same expected depth. With at most eight refinement steps per decision, AIC executes 3.3 steps on average across twelve benchmark tasks and lowers per-decision latency by 31–44% relative to a fixed eight-step policy on four control and robotics tasks. Its mean reward exceeds the best fixed-depth policy on seven tasks and trails it on four. Ablating adaptive stopping and value supervision separately shows that stopping accounts for most of the reward gain on PickCube and AntMaze, where it also executes more refinement steps and that supervision accounts for most of it on RareEvent. Project page: https://rohitsalla.github.io/aic-neurips2026/

1 Introduction

Neural policies typically use a fixed computation graph to map each state to an action distribution. Iterative policies instead repeatedly refine a representation of the current state before selecting an action, reusing one block so that computation grows without adding parameters [Ghugare et al., 2026]. These iterations process the current decision before the next environment transition; however, the number of effective iterations, or refinement depth, varies from one decision to the next. Routine motion may need little processing, whereas recovery from an unexpected contact benefits from additional iterations. When computation is costly, using the same depth everywhere spends too much on some states and too little on others, leading to the central question of how a policy can learn, at each state, whether further refinement is worth its cost.

Learned stopping provides a mechanism for making this allocation. ACT and PonderNet learn how many iterations to use for an input through prediction objectives and computation regularization [Graves, 2016, Banino et al., 2021]; learned metacontrol uses RL to choose internal computations [Hamrick et al., 2017]. For iterative policies, Ghugare et al. [2026] establish benefits of recurrent computation at fixed depth and POGP [Salla et al., 2026] stops diffusion denoising once predicted value gains become small; learning the stopping decision itself from environment return remains open. Here, stopping determines which action is executed and therefore affects subsequent states and rewards. Learning when to stop requires assigning credit to that decision jointly with learning how to act.

To address this problem, we introduce Adaptive Internal Computation (AIC), which makes action selection and stopping two decisions of the same policy. A halting head reads the current repre-

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

sentation and chooses whether to refine it further or use it to act now. Proximal policy optimization (PPO) updates the probabilities of both actions and stopping depths, while an expected-depth penalty charges for computation. Intermediate value predictions receive a shared rollout target as auxiliary supervision (Section 3). Our contribution has three parts. First, we formulate a joint distribution over stopping depth and action so that the environment return trains both decisions. Second, we show that when only some states benefit from further refinement, an ideal state-dependent policy outperforms every state-independent schedule at the same expected depth. Third, we evaluate AIC against fixed-depth and adaptive baselines, with a training-data control and an ablation of adaptive stopping and value supervision.

2 Related Work

AIC builds on methods that adapt computation to the input, on prior uses of extra computation inside policies and on early stopping in diffusion policies; we also draw on methods for attributing performance gains.

Adaptive computation. Adaptive-computation methods let a network vary how much it computes for each input. Adaptive Computation Time (ACT) adds a halting unit to a recurrent network and trains it through the task loss and a ponder cost [Graves, 2016], a mechanism that Universal Transformers apply per position [Dehghani et al., 2018]. PonderNet instead learns a conditional stopping probability at each step, trained through an expected prediction loss and a regularizer toward a geometric prior [Banino et al., 2021]. Early-exit networks attach classifiers to intermediate layers and stop once a prediction is confident [Teerapittayanon et al., 2016, Huang et al., 2017], while routing methods learn to skip individual layers or blocks for a given input [Wang et al., 2018b, Wu et al., 2018, Raposo et al., 2024]. AIC uses these established stopping primitives to select a depth-specific action policy. Its RL formulation includes both action and stopping probabilities in the PPO ratio, together with an expected-depth penalty and intermediate value supervision (Section E).

Computation in policies. Ghugare et al. [2026] studies how recurrent computation improves fixed-depth policies and identifies state-dependent depth within one policy as an open direction. In contrast, AIC learns that depth decision directly, examining the allocation opportunity at equal expected computation. While metareasoning and learned metacontrol already optimize computation choices [Hay et al., 2014, Hamrick et al., 2017], AIC applies this idea to refinement inside the action policy, without requiring explicit transition-model queries. Planning and search methods also spend additional computation on each decision, by expanding a search tree or unrolling a learned model [Tamar et al., 2016, Farquhar et al., 2017, Oh et al., 2017, Guez et al., 2018, Silver et al., 2017, Schrittwieser et al., 2020]; however, AIC instead refines a latent representation and queries no model.

Stopping in diffusion policies. POGP learns prefix values by bootstrapping along a diffusion denoising chain and stops when successive predicted gains are small [Salla et al., 2026], so its value predictions supply the stopping rule without a halting network. AIC instead trains a halting head through the joint policy ratio and a depth penalty; its per-depth critics are auxiliary and are not read at inference (Section A.10). Stopping therefore does not rely on value increments or on the diminishing-returns condition under which a first-crossing rule is optimal (Section C).

Attributing gains. Intermediate supervision can improve representations [Lee et al., 2015] and changing the policy can change the data it collects. Our ablation and data comparisons examine these effects, while early-exit interventions test sensitivity to truncation [Vig et al., 2020, Morcos et al., 2018] (Section 4.2).

3 Method: Adaptive Internal Computation

We consider a policy that interacts with an environment in discrete steps: at step t it observes a state s_t, executes an action a_t and receives a reward r_t and it aims to maximize expected discounted return. An iterative policy does not map s_t to an action in a single pass. It refines a latent representation of s_t over internal steps $d = 1, \ldots, d_{\max}$, during which the environment supplies no new observation or reward. Each internal step costs computation and the number of useful steps differs across states. The problem is to learn, together with the actions, a stopping depth $\tau_t \in \{1, \ldots, d_{\max}\}$

for each state that keeps return high and expected depth low. AIC treats (τ_t, a_t) as one decision of a joint policy, so the return that follows trains both choices (Figure 1).

Architecture. An encoder $f_{\text{enc}} : \mathcal{S} \to \mathbb{R}^{d_z}$ maps the state to $z_0 = f_{\text{enc}}(s)$. It uses a two-layer MLP for proprioception, a small CNN for grid states, or a frozen GPT-2 [Radford et al., 2019] for language inputs. A residual block then applies $z_d = z_{d-1} + \text{MLP}(z_{d-1})$ for $d = 1, \ldots, d_{\max}$, reusing its parameters at every depth. Three lightweight heads read each z_d: an action distribution π_d, a value prediction V_d and a conditional stopping probability $h_d \in [0, 1]$ for $d < d_{\max}$. Stopping at the cap is forced. The main configuration uses $d_z = 256$ and $d_{\max} = 8$ (Figure 1 and section A).

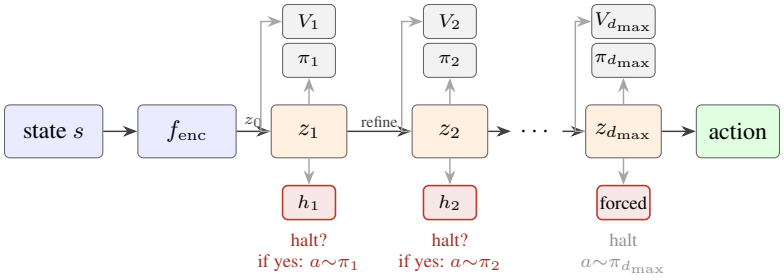


Figure 1: **AIC architecture.** The encoder produces z_0; a shared residual MLP refines z_d. Each depth has action and value heads, with a learned halting head before the forced cap. The action and halting heads learn through the joint policy objective; value heads provide auxiliary shared-return supervision.

Each z_d represents the current observation after d transformations. The heads read this representation directly, with value supervision updating the features shared by action selection and halting during training. The value outputs themselves are auxiliary: the halting head takes z_d as its input.

Executing one decision. The policy encodes s_t, computes z_1 and either stops to draw $a_t \sim \pi_1$ or refines again. This repeats until the selected depth or forced cap. It then sends one action to the environment; internal refinement supplies no new observations or intermediate actions.

Training and evaluation. Training samples Bernoulli stopping decisions and evaluates all depths for auxiliary value supervision. At evaluation, refinement ends at the first $d < d_{\max}$ with $h_d > 0.5$, or at the cap and an action is drawn from π_d. We measure evaluation return and depth under this threshold rule; Section A.12 explains its relationship to stochastic training.

3.1 Joint Policy and Training

For the return to train stopping as well as acting, the stopping depth must enter the policy's probability model. We therefore define a joint distribution over depth and action, built from the per-depth stopping probabilities. Let $h_{\theta,d}(s)$ be the probability of stopping upon reaching depth d, so that the probability of reaching d is $S_{\theta,d}(s) = \prod_{j<d}(1 - h_{\theta,j}(s))$, with an empty product of one. The stopping depth and action then follow

$$q_\theta(d \mid s) = \begin{cases} S_{\theta,d}(s) h_{\theta,d}(s), & d < d_{\max}, \\ S_{\theta,d_{\max}}(s), & d = d_{\max}, \end{cases} \qquad p_\theta(a, d \mid s) = q_\theta(d \mid s) \pi_{\theta,d}(a \mid z_{\theta,d}(s)). \tag{1}$$

The cap receives the probability of surviving every earlier stopping decision. This makes q_θ a normalized distribution and gives $\mathbb{E}[\tau \mid s] = \sum_d S_{\theta,d}(s)$ (Lemma 2).

The distinction between h_d and $q_\theta(d \mid s)$ matters: h_d is conditional on having reached depth d, whereas q_θ includes every earlier decision to continue. For example, stopping at depth two requires continuing at depth one and then stopping, so its probability is $(1-h_1)h_2$ when $2 < d_{\max}$. The joint log-probability consequently contains the continuation decisions, the final stopping decision when it is not forced and the selected action's log-probability.

For a sampled pair, a positive advantage means that the estimated return exceeds the rollout baseline. The actor objective favors increasing that pair's relative likelihood, while a negative advantage favors decreasing it, subject to PPO clipping. Reward credit therefore applies to the recorded continuation

decisions, stop and action together. The auxiliary value loss trains a separate prediction task on their shared representation.

Training objective. We train the joint policy with PPO, which compares the probability of each recorded decision under the current parameters with its probability under the frozen rollout parameters θ_{old}. Because a decision is the pair (a_t, τ_t), the ratio $\rho_t(\theta) = p_\theta(a_t, \tau_t \mid s_t)/p_{\theta_{\text{old}}}(a_t, \tau_t \mid s_t)$ includes the stopping probabilities. Writing $\widehat{\mathbb{E}}_t$ for the average over rollout samples and $\widehat{v}_{\theta,d}(s) = V_{\theta,d}(z_{\theta,d}(s))$ for the value prediction at depth d and holding rollout advantages and targets fixed, we minimize

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PPO}}(\theta) + \frac{\lambda_V}{d_{\max}} \sum_{d=1}^{d_{\max}} \widehat{\mathbb{E}}_t\big[(\widehat{v}_{\theta,d}(s_t) - V_t^{\text{targ}})^2\big] + \lambda_c \widehat{\mathbb{E}}_t\big[C_\theta(s_t)\big], \tag{2}$$

$$C_\theta(s) = \frac{1}{d_{\max}} \sum_{d=1}^{d_{\max}} d\, q_\theta(d \mid s) = \frac{1}{d_{\max}} \sum_{d=1}^{d_{\max}} S_{\theta,d}(s). \tag{3}$$

The cost gradient passes through the current stopping probabilities at the frozen rollout states. This defines a local PPO regularizer; Section A.7 distinguishes it from optimizing a trajectory return that includes future computation costs.

The cost term favors earlier stopping, while reward can favor continuing when the resulting action improves.

Shared value targets. Each intermediate representation is trained to predict the same realized GAE target [Schulman et al., 2015], even if its action head was not selected. The diagnostic

$$\widehat{\text{VoC}}(s, d) := \widehat{v}_{\theta,d}(s) - \widehat{v}_{\theta,d-1}(s), \qquad d = 2, \ldots, d_{\max}, \tag{4}$$

therefore compares predictions of a shared target. The quantity $\widehat{\text{VoC}}$ is a diagnostic of the value heads; halting is trained through the joint actor and cost gradients. Proposition 3 characterizes the shared-target predictions and Section A.10 shows why replacing their outputs leaves a frozen stopping policy unchanged. Advantages use a state-only baseline that does not condition on the sampled depth; Algorithm 1 and Section A.7 give the full training procedure.

3.2 Allocation Bound

To study the opportunity for allocation, consider the return from choosing an action at each depth while holding future behavior fixed. For depth-specific action distributions and a common continuation policy $\bar{\pi}$, define $W_d(s) = \mathbb{E}_{a \sim \pi_d(\cdot|s)}[Q^{\bar{\pi}}(s, a)]$. An ideal allocator maximizes $W_d(s) - cd$. Stopping at the first increment below cost is optimal when $W_{d+1} - W_d$ is non-increasing; otherwise a later gain can justify continuing through an unhelpful step (Section C). The matching cost units are $c = \lambda_c / d_{\max}$; this ideal problem is distinct from the PPO surrogate and shared-target critics.

The following result formalizes why varying depth across states can help at a common expected budget.

Theorem 1 (Benefit of state-dependent allocation)**.** *Fix a state distribution μ, a depth cap $d_{\max} \geq K \geq 2$ and $0 < \epsilon < 1$. Easy states of mass $1 - \epsilon$ attain reference value W^* at every depth; hard states attain it at depths $d \geq K$ and have shortfall at least $\Delta(1 - d/K)$ for $d < K$, where $\Delta > 0$. An ideal allocator attains W^* with expected depth $b = 1 + \epsilon(K - 1) < K$. Every state-independent stopping distribution with $\mathbb{E}[\tau] \leq b$ has expected shortfall at least*

$$\mathcal{R}_\mu \geq \epsilon \Delta (1 - b/K) > 0. \tag{5}$$

The allocator reserves depth K for hard states, while every state-independent schedule at the same expected budget has positive shortfall, including randomized schedules. Section B gives the proof, extensions and sequential-return conditions. The result concerns an ideal allocator that knows the easy/hard partition, with the state distribution and action policies held fixed. Learning must recover a useful allocation from sampled outcomes. For sequential tasks, Section B.5 relates the shortfall to return using the state distribution induced by each policy.

Table 1: **Benchmark comparison at nominal B=8.** Δ: AIC minus the per-budget fixed baseline, whose selected training depths appear in parentheses with seed counts as superscripts. Gap recovery is measured toward the privileged reference and is undefined when the reference does not exceed the fixed baseline. Best fixed: strongest fixed baseline at any budget $B' \leq 8$ in Table 14, with B' in parentheses; Δ_{best} is AIC minus that value. *: $p < .05$, two-sided Welch's t-test over five seeds with the standard deviations of Table 14.

Task	Fixed (K_{train})	AIC	Privileged	Δ	%Δ	Gap recovery	Best fixed (B')	Δ_{best}
MiniGrid-DistShift2 (sat. ctrl.)	1.0000 (1)	1.0000	1.0000	.0000	0.00%	—	1.0000 (1)	.0000
HalfCheetah-v4	.9329 (8)	.9348	.9744	$+.0019^*$	+0.20%	4.6%	.9329 (8)	$+.0019^*$
GPT-2 on GLUE-RTE	.8311 (8)	.8334	.9323	+.0023	+0.28%	2.3%	.8311 (8)	+.0023
OGBench cube-double-play-v0	.4281 (8)	.4299	.5646	+.0018	+0.42%	1.3%	.4281 (8)	+.0018
AntMaze (low-het.)	.6704 (8)	.6776	.8685	$+.0072^*$	+1.07%	3.6%	.6916 (1)	$-.0140^*$
OGBench scene-play-v0	.4075 (8)	.4167	.5565	$+.0092^*$	+2.26%	6.2%	.4075 (8)	$+.0092^*$
ManiSkill StackCube-v1	.4458 (8)	.4579	.6274	$+.0121^*$	+2.71%	6.7%	.4508 (4)	+.0071
ManiSkill PickCube-v1	.4741 (8)	.4926	.6756	$+.0185^*$	+3.90%	9.2%	.4911 (4)	+.0015
AntMaze (main)	.5018 ($2^{(1)}, 8^{(4)}$)	.5209	.7367	$+.0191^*$	+3.81%	8.1%	.5095 (4)	$+.0114^*$
RareEvent-StackCube-Misalign	.5575 ($1^{(2)}, 8^{(3)}$)	.5918	.7523	$+.0343^*$	$+6.15\%^\dagger$	$17.6\%^\dagger$	.6338 (1)	$-.0420^*$
OGBench puzzle-4x4-play-v0	.4069 (8)	.4022	.5185	−.0047	−1.16%	−4.2%	.4069 (8)	−.0047
MiniGrid-FourRooms	.9730 (8)	.9514	.9730	$-.0216^*$	−2.22%	—	.9730 (8)	$-.0216^*$

† On RareEvent-StackCube-Misalign, the stopping and supervision ablation attributes most of the gain to value supervision (Section F).

4 Experiments

Our experiments address three questions. First, does AIC reach higher reward or lower computation than fixed-depth policies under the same cap on refinement steps (Section 4.1)? Second, do its gains come from where it spends computation, from the data it collects, or from value supervision (Section 4.2)? Third, how does it compare with other learned halting methods (Section 4.3)? We first describe the tasks, baselines and computation budgets shared by all three.

Tasks and baselines. The twelve-task benchmark covers continuous control, manipulation, navigation, language, gridworld and rare events (Section D). GPT-2/GLUE-RTE is a single-step entailment-classification task (Section D.1). Fixed-depth baselines share AIC's encoder, refinement block, per-depth heads and PPO [Schulman et al., 2017] objective but always stop at a training depth $K_{\text{train}} \in \{1, 2, 4, 8\}$. A privileged reference selects depth using simulator-side difficulty estimates $\hat{\kappa}(s)$ (Section J). Hyperparameters, seeds and loss-weight sweeps appear in Section I.

Evaluation protocol. Benchmark results are five-seed means with standard deviations (Table 14); Table 1 lists the $B = 8$ means. Each task keeps its own reward scale, so we compare methods within tasks and the POGP comparison uses POGP's return scale. The data-source ablation, the stopping and supervision ablation and the memory, transfer and sensitivity studies use their own training runs and are not pooled with the benchmark, whereas the truncation ablation and the halting-baseline comparison reuse the benchmark checkpoints.

Computation budgets. The nominal budgets $B \in \{1, 2, 4, 8\}$ cap executed depth: a fixed checkpoint uses $\min(K_{\text{train}}, B)$ refinements. Fixed baselines come from a separate training sweep at each budget, with learning rate and entropy coefficient tuned on held-out validation episodes at that budget. For each seed, we select the checkpoint with the highest validation return at budget B, evaluating each candidate with $\min(K_{\text{train}}, B)$ refinement steps. Parentheses list the selected training depths, with seed counts when they differ (Section K). AIC at $B < 8$ is retrained with $d_{\max} = B$ and the default hyperparameters, so at $B = 1$ it has no stopping decision and the $B = 1$ differences reflect training differences unrelated to allocation. The main comparison places both methods under the same execution cap; Table 15 gives AIC's realized depth at every budget and Table 10 every method's depth at $B = 8$, while the sensitivity study in Section I matches mean depth instead.

4.1 Comparison to Fixed-Depth Methods

A higher return can come from state-dependent stopping, intermediate value supervision, changes in the learned action policies, or different training trajectories. We therefore ask what reward and computation the complete policy achieves relative to fixed and adaptive alternatives; how performance changes when we vary the data collector, the learner, the stopping configuration or the auxiliary supervision; and whether the frozen policy depends on refinement at the states where it selects greater

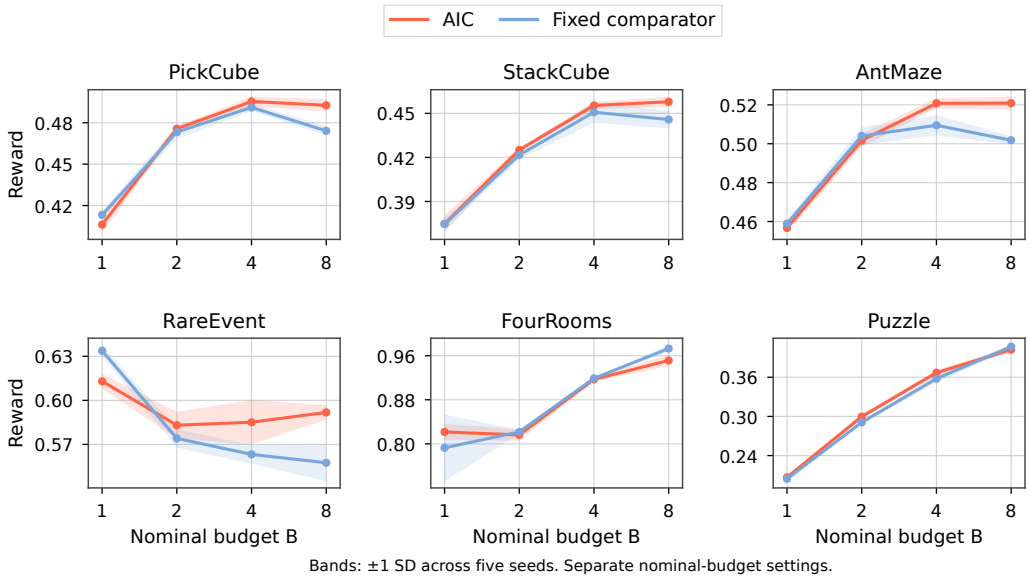


Figure 2: **Performance across nominal budgets on six tasks.** Points are five-seed means with ± 1 standard-deviation bands (Table 14); each point is a separate budget setting and realized depth can differ between methods. RareEvent peaks at $B = 1$ for both methods. Figure 4 shows the remaining six tasks.

Table 2: **POGP–AIC comparison with an eight-step cap.** Undiscounted return over 100-step episodes (Section D.2) and iterations per action, mean $\pm$ standard deviation over five seeds.

	POGP		AIC	
Environment	Return ↑	Steps ↓	Return ↑	Steps ↓
HalfCheetah	330 ± 22	$5.8 \pm .3$	325 ± 25	$4.2 \pm .3$
Walker2d	180 ± 15	$6.2 \pm .3$	183 ± 17	$4.8 \pm .4$
Ant	91 ± 10	$6.8 \pm .4$	88 ± 12	$5.3 \pm .4$
Hopper	205 ± 30	$5.9 \pm .3$	208 ± 28	$4.5 \pm .3$

depth. The benchmark answers the first question, the data-source ablation and the stopping and supervision ablation answer the second and the truncation ablation answers the third, with an equal-work control that separates which states are truncated from how much computation is removed.

Fixed-depth results. At $B = 8$, AIC exceeds the per-budget fixed baseline on nine tasks, seven of them at $p < .05$ (two-sided Welch's t-test over five seeds), ties on DistShift and trails on FourRooms and Puzzle (Table 1). Against the strongest fixed baseline at any budget within the cap, AIC leads on seven tasks, three at $p < .05$, ties on one and trails on four: depth-one baselines lead on RareEvent (.6338) and low-heterogeneity AntMaze (.6916) and the PickCube margin narrows to $+.0015$ against Fixed-4, whose depth is closest to AIC's 3.47. Each budget has its own tuned sweep, so a per-budget baseline can trail another budget's: on PickCube, the $B = 8$ sweep selects $K_{\text{train}} = 8$ (.4741), while the separately tuned $B = 4$ sweep's Fixed-4 reaches .4911. The largest per-budget gain is on RareEvent, where the stopping and supervision ablation attributes most of the change to value supervision and the privileged reference measures the remaining headroom relative to task-specific depth heuristics (Section J).

Comparison with POGP. Table 2 compares AIC with POGP on four MuJoCo locomotion tasks under an eight-step cap. AIC uses 22–28% fewer iterations per action on all four, while its mean return stays within 3–5 of POGP's, less than 0.3 standard deviations. Both methods use POGP's diffusion backbone, so one iteration is one denoising step in each (Section D.2).

Table 3: **Sensitivity to targeted and random truncation.** Targeted: decisions at or above the 75th-percentile depth, forced to d=1. Random: count-matched random early exits. Targeted drops exceed random drops on every non-saturated task, including the final two rows; ratios are omitted when the random drop is non-positive. Table 16 gives selected fractions, removed refinements and a random control matched on removed work.

Task	Normal	Targeted Drop	Random Drop	Targeted−Random	Ratio
MiniGrid-DistShift2 (sat.)	1.0000	.0000	.0000	.0000	—
AntMaze (low-het.)	.6776	.0169	−.0033	.0202	—
HalfCheetah-v4	.9348	.0247	.0108	.0139	2.3×
GPT-2 on GLUE-RTE	.8334	.0501	.0142	.0359	3.5×
AntMaze (main)	.5209	.0845	.0228	.0617	3.7×
ManiSkill PickCube-v1	.4926	.1062	.0386	.0676	2.8×
ManiSkill StackCube-v1	.4579	.1148	.0446	.0702	2.6×
OGBench cube-double-play-v0	.4299	.2091	.1399	.0692	1.5×
OGBench scene-play-v0	.4167	.1945	.1211	.0734	1.6×
RareEvent-StackCube-Misalign	.5918	.0034	−.0016	.0050	—
MiniGrid-FourRooms	.9514	.1189	.0396	.0793	3.0×
OGBench puzzle-4x4-play-v0	.4022	.2105	.1553	.0552	1.4×
mean over 9 positive comparisons		.0894	.0430	.0463	—

4.2 Ablations and Controls

AIC differs from a fixed-depth policy in three ways: (1) it decides at each state how many refinement steps to run before acting (Section 4.2.1); (2) it collects different training data, because a policy that acts differently visits different states (Section 4.2.2); and (3) it trains a value head at every depth, which shapes the representation shared with the action and halting heads (Section 4.2.3).

4.2.1 Truncation Study

We first ask whether the frozen policy relies more on refinement at the states where it chooses greater depth. The targeted intervention forces high-depth decisions to stop at depth one and the control truncates an equally sized random subset, so a larger targeted drop indicates greater reliance on refinement at the selected states. Matching the number of interventions does not match removed computation: truncating depth eight removes seven steps, while truncating depth two removes one. For a reference set of decisions $\mathcal{I}$, forcing $A \subseteq \mathcal{I}$ to depth one removes $M(A) = \sum_{i \in A}(\tau_i - 1)$ refinements, so sets of equal size can differ in $M(A)$. We therefore also run an equal-work control (Section L).

Across the nine tasks where AIC beats its baseline, targeted truncation lowers reward by 0.0894 on average and random truncation by 0.0430, a difference of 0.0463. Part of this difference reflects removed work, because targeted decisions sit at greater depth: 4.80 versus 2.47 removed refinements per intervened decision on PickCube (Table 16). Matching removed refinements reduces the mean difference to 0.0118, yet the targeted drop still exceeds the equal-work drop by .0065–.0219 on PickCube, StackCube, both AntMaze variants, RTE, RareEvent and FourRooms; on HalfCheetah, Cube, Scene and Puzzle the two agree within $.004$. RareEvent shows small truncation effects, consistent with its large supervision effect in the stopping and supervision ablation. FourRooms and Puzzle show positive intervention gaps despite trailing their fixed baselines, so reliance on refinement and improvement over a separately trained baseline are distinct properties of a policy.

4.2.2 Data-Source Ablation

When training a policy with RL, a policy that acts differently will inevitably see different data. To separate the data from the policy, we swap them on ManiSkill PickCube-v1 and ManiSkill StackCube-v1. A Fixed-1 policy and AIC at $B = 8$ are each trained, frozen and used to collect one fixed-size dataset used to train a new Fixed-1 and AIC learners from scratch through behavioral cloning and value distillation.

Changing only the collector or only the learner adds a reward of 0.03–0.04, while changing both adds 0.16–0.17 (Table 4). The combined gain exceeds the sum of the two separate gains by $.0981$ on PickCube and $.0850$ on StackCube, with standard errors of $.015$ and $.018$. AIC's gain therefore

Table 4: **Data-source ablation.** Rows are collector–learner pairs; columns give reward and the learner's mean halting depth $\mathbb{E}$[steps]. The combined gain exceeds the sum of the separate gains. Values are means $\pm$ standard deviation over five learner seeds, each collector produced one dataset.

	ManiSkill PickCube-v1		ManiSkill StackCube-v1	
Configuration	Reward	$\mathbb{E}$[steps]	Reward	$\mathbb{E}$[steps]
Fixed-1 collector → Fixed-1 learner	$.3195 \pm .025$	$1.00 \pm .00$	$.2854 \pm .030$	$1.00 \pm .00$
Fixed-1 collector → AIC learner	$.3565 \pm .018$	$1.00 \pm .00$	$.3194 \pm .022$	$1.00 \pm .00$
AIC collector → Fixed-1 learner	$.3522 \pm .013$	$1.00 \pm .00$	$.3241 \pm .014$	$1.00 \pm .00$
AIC collector → AIC learner	$.4873 \pm .003$	$3.34 \pm .06$	$.4431 \pm .002$	$3.99 \pm .08$

Table 5: **Stopping and Supervision Ablation.** Each configuration gives reward mean $\pm$ standard deviation over five seeds and mean executed depth $\bar{d}$ for an independently trained configuration. The non-adaptive configurations execute a fixed schedule of one refinement per decision. the adaptive configurations use AIC's learned stopping with $d_{\max} = 8$.

	Neither		Supervision only		Adaptive stopping only		Full AIC	
Task	Reward	$\bar{d}$	Reward	$\bar{d}$	Reward	$\bar{d}$	Reward	$\bar{d}$
PickCube	$.421 \pm .006$	1.00	$.434 \pm .005$	1.00	$.472 \pm .004$	3.61	$.488 \pm .004$	3.52
AntMaze	$.451 \pm .005$	1.00	$.458 \pm .004$	1.00	$.505 \pm .004$	3.28	$.518 \pm .003$	3.19
Memory	$.503 \pm .007$	1.00	$.515 \pm .006$	1.00	$.552 \pm .006$	3.42	$.568 \pm .005$	3.33
DoorKey	$.486 \pm .007$	1.00	$.499 \pm .006$	1.00	$.536 \pm .005$	3.49	$.551 \pm .005$	3.38
StackCube	$.394 \pm .006$	1.00	$.413 \pm .006$	1.00	$.441 \pm .005$	3.94	$.455 \pm .004$	3.81
RareEvent	$.558 \pm .008$	1.00	$.589 \pm .006$	1.00	$.560 \pm .009$	1.92	$.591 \pm .006$	1.84

does not come from its training data alone, since most of it appears only when an AIC learner is trained on AIC data, the one setting that also refines beyond depth one.

4.2.3 Stopping and Supervision Ablation

AIC adds two components to a one-step policy, adaptive stopping and value supervision at every depth, which we will isolate in this experiment. We train four configurations independently, with each component switched on or off (Table 5). Let R_{as} denote reward with adaptive stopping $a \in \{0, 1\}$ and auxiliary supervision $s \in \{0, 1\}$. The stopping effects $R_{10} - R_{00}$ and $R_{11} - R_{01}$ measure the reward change from enabling stopping without and with supervision. The supervision effects $R_{01} - R_{00}$ and $R_{11} - R_{10}$ measure the change from enabling supervision without and with stopping.

On PickCube the stopping effects are .051 and .054 and the supervision effects .013 and .016; on AntMaze they are .054 and .060 versus .007 and .013. RareEvent reverses the pattern: stopping changes reward by .002 at either supervision setting, while supervision changes it by .031. StackCube gains from both factors and the memory tasks Memory and DoorKey follow PickCube (Section P). Because the non-adaptive configurations execute one refinement per decision and the adaptive configurations execute 1.8–3.9, each stopping effect combines state-dependent allocation with additional computation. The Fixed-4 baseline of Table 14, trained separately, reaches .4911 on PickCube and .5095 on AntMaze, above the corresponding adaptive-stopping-only configurations. Appendix F outlines a comparison at matched average computation that could isolate the effect of allocation.

4.3 Baseline Comparisons with Other Halting Methods

We next compare AIC with other stopping methods on seven tasks: four untrained rules (random stopping, confidence thresholding, action agreement and a value-gain threshold) and two jointly trained methods, ACT-ponder [Graves, 2016] and a learned non-VoC threshold. Section G defines all methods and Table 10 gives their mean depths.

AIC's reward is close to that of the trained baselines, exceeding ACT-ponder by .0016 on PickCube and learned non-VoC by .0001 on AntMaze, both within one standard error of the difference. Across the seven tasks, AIC is within two standard errors of both trained baselines except on StackCube, where AIC is higher and FourRooms, where ACT-ponder is higher. AIC uses the lowest mean depth of the three on all seven tasks (Figure 3); on FourRooms that lower depth comes with lower reward.

Table 6: **Adaptive halting baselines at $B{=}8$ across 7 tasks.** Mean reward over five seeds, with the seed standard deviation as a subscript; AIC and Fixed repeat Table 14. The trained methods reach similar rewards on several tasks at different mean depths. ACT-ponder leads on FourRooms and the random schedule, which executes one refinement there, leads on RareEvent.

	Trained jointly with the policy			Untrained stopping rules				
Task	AIC	ACT-ponder	Lrn non-VoC	Conf. thr.	Action-agr.	Val.-gain thr.	Random	Fixed
PickCube	$.4926_{\pm.0045}$	$.4910_{\pm.0041}$	$.4907_{\pm.0047}$	$.4523_{\pm.0062}$	$.4589_{\pm.0057}$	$.4425_{\pm.0066}$	$.4564_{\pm.0049}$	$.4741_{\pm.0034}$
StackCube	$.4579_{\pm.0036}$	$.4521_{\pm.0039}$	$.4527_{\pm.0042}$	$.4218_{\pm.0058}$	$.4378_{\pm.0061}$	$.4066_{\pm.0072}$	$.4163_{\pm.0053}$	$.4458_{\pm.0060}$
AntMaze (main)	$.5209_{\pm.0034}$	$.5200_{\pm.0031}$	$.5208_{\pm.0036}$	$.4626_{\pm.0049}$	$.4972_{\pm.0044}$	$.4680_{\pm.0058}$	$.4824_{\pm.0047}$	$.5018_{\pm.0022}$
GLUE-RTE	$.8334_{\pm.0009}$	$.8300_{\pm.0042}$	$.8299_{\pm.0040}$	$.7617_{\pm.0031}$	$.8110_{\pm.0026}$	$.7957_{\pm.0029}$	$.7932_{\pm.0034}$	$.8311_{\pm.0024}$
Scene	$.4167_{\pm.0048}$	$.4130_{\pm.0044}$	$.4136_{\pm.0051}$	$.3884_{\pm.0055}$	$.3530_{\pm.0063}$	$.3716_{\pm.0059}$	$.3903_{\pm.0052}$	$.4075_{\pm.0044}$
RareEvent	$.5918_{\pm.0050}$	$.5868_{\pm.0058}$	$.5870_{\pm.0055}$	$.5797_{\pm.0071}$	$.4198_{\pm.0098}$	$.4198_{\pm.0098}$	$.6178_{\pm.0046}$	$.5575_{\pm.0125}$
FourRooms	$.9514_{\pm.0103}$	$.9658_{\pm.0087}$	$.9586_{\pm.0096}$	$.8829_{\pm.0134}$	$.7748_{\pm.0182}$	$.7315_{\pm.0191}$	$.7045_{\pm.0213}$	$.9730_{\pm.0064}$

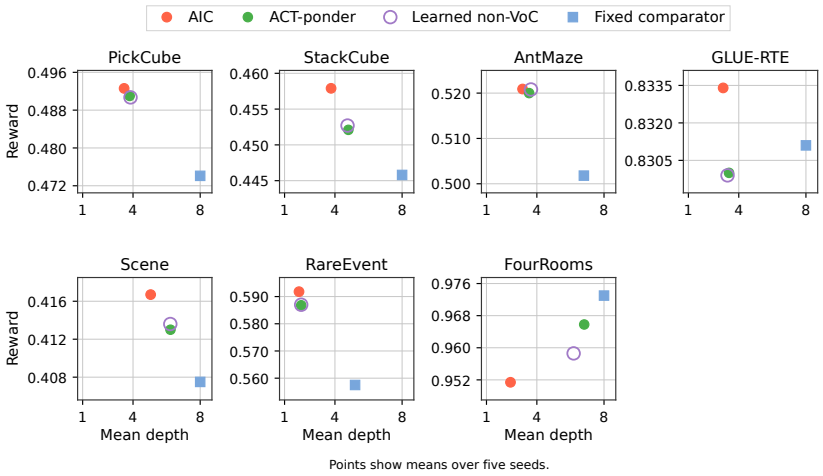


Figure 3: **Reward versus mean halting depth at nominal $B = 8$.** Panels show AIC, ACT-ponder, learned non-VoC and fixed baselines, using Tables 6 and 10. A fixed baseline's depth is the seed average of $\min(K_{\text{train}}, 8)$ over its selected checkpoints.

The learned non-VoC baseline shares AIC's encoder, refinement block, action heads and value heads and differs only in how its halting head is trained: its stopping probabilities neither enter the PPO ratio nor receive advantage credit and the head instead learns from a stop-gradient copy of z_d to predict whether the next refinement leaves the action distribution unchanged. The comparison therefore separates return-based stopping credit from a learned convergence signal on the same representation (Table 9). Parameter counts are 2.071M for AIC, 2.069M for ACT-ponder and 2.110M for learned non-VoC. On GPT-2/RTE, with a checkpoint trained at $\lambda_c = .003$ and mean depth 4.08, AIC executes 49% fewer refinements than Fixed-8 and lowers end-to-end latency by 7.8%, since the frozen encoder dominates the total cost. Training evaluates every depth for value supervision. On PickCube, AIC training took $1.22\times$ as long as Fixed-8 training (Section Q). Overall, training the halting head through the return matches or exceeds the reward of the other learned halting methods on six of seven tasks while executing fewer refinement steps on all seven.

5 Discussion

Our analysis identifies three ingredients for useful allocation: state-dependent benefits from refinement, performance headroom and a representation from which the allocator can recognize those benefits. The tasks differ in where the gains come from. On PickCube and AntMaze, adaptive stopping, which also executes more refinements, contributes more than supervision; on RareEvent, supervision contributes more.

FourRooms is the clearest failure: AIC trails its mean-depth-matched baseline at all twenty loss-weight settings, by 0.7 to 2.1 percentage points of normalized reward (Table 11; best $-.7 \pm .3$ at

Table 7: **Transfer to five diffusion-based policy families at nominal $B = 8$.** Gain $\Delta = R_{\mathrm{AIC}} - R_{\mathrm{Fixed}}$ with its standard error over five seeds as a subscript. Fixed executes eight refinement steps per action in every family; Table 21 lists the returns and AIC's mean halting depth.

Task	DiffusionQL	DPPO	D3P	DSAC-D	IDQL
HalfCheetah-v4	$+.002_{\pm.003}$	$+.001_{\pm.002}$	$+.002_{\pm.003}$	$+.001_{\pm.003}$	$+.001_{\pm.002}$
ManiSkill PickCube-v1	$+.020_{\pm.003}$	$+.018_{\pm.003}$	$+.023_{\pm.003}$	$+.010_{\pm.004}$	$+.012_{\pm.003}$
ManiSkill StackCube-v1	$+.015_{\pm.003}$	$+.013_{\pm.003}$	$+.018_{\pm.003}$	$+.008_{\pm.003}$	$+.010_{\pm.003}$
OGBench scene-play-v0	$+.012_{\pm.003}$	$+.008_{\pm.003}$	$+.014_{\pm.003}$	$+.004_{\pm.003}$	$+.010_{\pm.003}$
AntMaze (main)	$+.017_{\pm.003}$	$+.015_{\pm.002}$	$+.018_{\pm.003}$	$+.010_{\pm.003}$	$+.014_{\pm.003}$
OGBench puzzle-4x4-play-v0	$-.003_{\pm.003}$	$-.005_{\pm.003}$	$+.002_{\pm.003}$	$-.006_{\pm.002}$	$+.001_{\pm.003}$
MiniGrid-FourRooms	$-.015_{\pm.005}$	$-.018_{\pm.005}$	$-.012_{\pm.004}$	$-.020_{\pm.004}$	$-.014_{\pm.004}$
RareEvent-StackCube-Misalign	$+.025_{\pm.005}$	$+.022_{\pm.005}$	$+.030_{\pm.005}$	$+.012_{\pm.005}$	$+.018_{\pm.005}$

$(\lambda_c, \lambda_V) = (.001, .5)$). Puzzle, in contrast, turns positive with stronger value supervision (Table 12), so its default-setting loss depends on the training configuration. Stopping and the action policies at intermediate depths must therefore be evaluated together.

Attached to DiffusionQL, DPPO, D3P, DSAC-D and IDQL (Table 7), AIC raises mean return on six of eight tasks in all five families, lowers it on FourRooms in all five and changes sign across families on Puzzle. HalfCheetah gains lie within one standard error in every family, gains on the other five tasks exceed two standard errors except DSAC-D on Scene and every FourRooms decrease exceeds two standard errors. Section R defines each family's refinement step and intermediate-action decoder and Section P extends AIC to memory tasks.

Limitations and future work. While our controls cannot fully separate where computation is spent from how much is spent, they show that the policy depends more on refinement at the states where it chooses greater depth, a difference that persists on seven tasks when the removed computation is matched. Reassigning recorded depths across states at fixed total computation would isolate this state–depth association further. Adaptive stopping does not help on every task and FourRooms remains below a fixed depth across the loss weights we tried on eleven of twelve tasks. however, against the fixed baseline tuned at the same eight-step budget, AIC had higher mean reward on nine tasks, tied on one, and had lower mean reward on two, while averaging 3.3 refinement steps. Our experiments use small refinement modules in simulation, yet the gains transfer to five diffusion-policy families, which suggests the approach is not tied to one architecture. For future work, we plan on scaling AIC to large backbones and real robots.

6 Conclusion

We introduced AIC, an iterative policy that treats stopping depth and action as one decision, so that environment return trains both. We showed that when only some states benefit from further refinement, an ideal state-dependent policy outperforms every state-independent schedule at the same expected depth. On twelve benchmark tasks with at most eight refinement steps per decision, AIC executes 3.3 steps on average; its mean reward exceeds the best fixed-depth policy on seven tasks and trails it on four. It also uses fewer steps than two jointly trained halting methods on all seven compared tasks and than POGP on four locomotion tasks, at comparable reward. Ablating adaptive stopping and value supervision separately shows that stopping accounts for most of the reward gain on PickCube and AntMaze and supervision on RareEvent. The adaptive runs also execute more refinement steps. Isolating the effect of allocation would require the work-matched comparison outlined in Appendix F. Learning when to think thus becomes part of learning how to act.

Acknowledgements The authors acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper. URL: `https://arc.vt.edu/`

References

A. Banino, J. Balaguer, and C. Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. 2018.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

G. Farquhar, T. Rocktäschel, M. Igl, and S. Whiteson. Treeqn and atreec: Differentiable tree-structured models for deep reinforcement learning. *arXiv preprint arXiv:1710.11417*, 2017.

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

R. Ghugare, M. Bortkiewicz, A. Ziarko, and B. Eysenbach. On the role of computation in reinforcement learning. *arXiv preprint arXiv:2602.05999*, 2026.

A. Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

A. Guez, T. Weber, I. Antonoglou, K. Simonyan, O. Vinyals, D. Wierstra, R. Munos, and D. Silver. Learning to search with mctsnets. In *International conference on machine learning*, pages 1822–1831. PMLR, 2018.

J. B. Hamrick, A. J. Ballard, R. Pascanu, O. Vinyals, N. Heess, and P. W. Battaglia. Metacontrol for adaptive imagination-based optimization, 2017. URL `https://arxiv.org/abs/1705.02670`.

P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

N. Hay, S. Russell, D. Tolpin, and S. E. Shimony. Selecting computations: Theory and applications. *arXiv preprint arXiv:1408.2048*, 2014.

G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, and K. Q. Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. Pmlr, 2015.

T. Liu, Y. Wang, X. Song, W. Zou, L. Chen, L. Wang, B. Shuai, J. Duan, and S. E. Li. Distributional soft actor-critic with diffusion policy. In *2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3038–3045. IEEE, 2025.

A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

J. Oh, S. Singh, and H. Lee. Value prediction network. *Advances in neural information processing systems*, 30, 2017.

S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pages 94937–94982, 2025.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019. URL `https://api.semanticscholar.org/CorpusID:160025533`.

D. Raposo, S. Ritter, B. Richards, T. Lillicrap, P. C. Humphreys, and A. Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

A. Ren, J. Lidard, L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pages 77288–77329, 2025.

R. K. Salla, M. Saravanan, and S. Stepputtis. Learning when to stop: Prefix-optimal dynamic diffusion policies for continuous control. *Reinforcement Learning Journal*, 7, 2026.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, Dec. 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, V. N. Rajesh, Y. W. Choi, Y.-R. Chen, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai, 2025. URL https://arxiv.org/abs/2410.00425.

S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

J. Vig, S. Gehrmann, Y. Belinkov, S. Qian, D. Nevo, Y. Singer, and S. Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355, 2018a.

X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *European Conference on Computer Vision*, pages 420–436. Springer, 2018b.

Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris. Blockdrop: Dynamic inference paths in residual networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8817–8826. IEEE, 2018.

S.-A. Yu, F. Gao, Y. Wu, C. Yu, and Y. Wang. D3p: Dynamic denoising diffusion policy via reinforcement learning. *arXiv preprint arXiv:2508.06804*, 2025.

Supplementary Material

The supplement begins with the method and its derivations, followed by the allocation theorem and the ideal stopping problem. Section A follows one decision from encoding to action, derives stopping probabilities and the PPO update and explains the role of each output head. Section B gives a concrete allocation example and a step-by-step proof of Theorem 1, followed by its extensions and sequential-return conditions. Section C derives optimal stopping for a known depth-value profile and identifies when a one-step threshold is sufficient.

Section D then defines the tasks. The empirical sections provide the prior-work comparison, factorial interpretation, adaptive baselines, computation accounting and sensitivity studies (Sections E to I). The remaining sections develop the privileged reference, full budget results, intervention and annotation procedures, allocation profiles and task-level interpretation (Sections J to N). Sections O to S collect the limitations, memory extensions, timing measurements, transfer details and a protocol summary.

A Method Details

The central distinction is between an environment step, which produces a new observation and reward and a refinement step, which processes the current observation again.

A.1 Environment steps and internal computation

At step t, the state is s_t, the action is a_t, the reward is r_t and the next state is s_{t+1}. The policy's parameters, denoted by θ, are the neural-network weights that training adjusts.

The discounted return from step t is the sum $G_t = \sum_{k\geq 0} \gamma^k r_{t+k}$, stopped at the end of a finite episode when applicable. For continuing tasks with bounded rewards, $0 \leq \gamma < 1$ ensures a finite return; bounded-horizon episodes can also use $\gamma = 1$. A value prediction estimates this return.

AIC adds a second index, d, for computation within one environment step. The policy encodes s_t, refines that encoding and decides whether to continue refining before it sends an action to the environment. The state, reward clock and available observation remain fixed throughout this internal sequence. Thus $d = 3$ means three refinements of the current decision, not three environment transitions. In this formulation, discounting applies across environment steps and the separate depth penalty charges for internal computation.

For a fully observed Markov decision process, s_t contains the information needed to predict the next-state distribution given an action. In a partially observed task, the policy instead needs a suitable representation of its observation history. The notation s_t can represent the information supplied to that policy, including its history representation. The memory-task description in Section P therefore distinguishes observation history from internal refinement.

A.2 Encoding, refinement and the three outputs

An encoder converts the input into a vector of numbers, $z_0 = f_{\text{enc}}(s)$. The encoder depends on the input type. A multilayer perceptron (MLP) processes proprioceptive inputs. A convolutional neural network (CNN) processes grid inputs and the language task uses the frozen GPT-2 representation described in Section D.1.

The refinement block repeatedly transforms the encoded vector:

$$z_d = z_{d-1} + F_\theta(z_{d-1}), \qquad d = 1, \ldots, d_{\max}.$$

Here F_θ is the shared MLP block. The same block parameters are used at every depth, so increasing depth repeats computation without allocating a different refinement network to each iteration. All representations describe the current decision; their action quality can vary non-monotonically with depth.

Three lightweight heads read the representation at a depth. The action head $\pi_{\theta,d}$ gives the distribution used if that depth is selected. For a discrete action space it assigns probabilities to actions; for a continuous action space its likelihood is a probability density. The value head $V_{\theta,d}$ produces

a scalar prediction of the shared training target. The halting head produces $h_{\theta,d} \in [0,1]$, the probability of stopping if depth d has been reached. Only depths below $d_{\max}$ require a learned stopping probability, because stopping at the cap is compulsory.

The main configuration uses latent dimension 256, cap eight and depth-specific heads. Throughout the appendix, $\pi_{\theta,d}(a \mid s)$ abbreviates $\pi_{\theta,d}(a \mid z_{\theta,d}(s))$ and $\widehat{v}_{\theta,d}(s) = V_{\theta,d}(z_{\theta,d}(s))$. The hats distinguish learned value predictions from the depth-specific returns used in the allocation analysis.

A.3 Stopping-distribution example

Consider one state with a cap of four solely to make the calculation short. Suppose its three conditional stopping probabilities are $h_1 = .2$, $h_2 = .6$ and $h_3 = .25$. At depth one, the policy stops with probability .2 and continues with probability .8. The probability of stopping at depth two is therefore $.8 \times .6 = .48$, not .6: the policy must first reach that depth. Continuing through the first two depths has probability $.8 \times .4 = .32$. Of that remaining mass, $.32 \times .25 = .08$ stops at depth three and $.32 \times .75 = .24$ reaches the forced cap. Consequently,

$$(S_1, S_2, S_3, S_4) = (1, .8, .32, .24), \qquad (q_1, q_2, q_3, q_4) = (.2, .48, .08, .24).$$

The entries of q sum to one. The survival probability S_d answers whether depth d is reached; q_d answers whether it is the final depth. Conditional stopping probabilities h_d need not sum to one, because each applies to a different conditional event.

The expected depth can be calculated in either of two equivalent ways:

$$\mathbb{E}[\tau \mid s] = 1(.2) + 2(.48) + 3(.08) + 4(.24) = 1 + .8 + .32 + .24 = 2.36.$$

The first expression weights each final depth by its probability. The second counts one unit of computation for every depth reached: the first is always reached, the second with probability .8 and so on. Normalizing by the cap gives $C(s) = 2.36/4 = .59$. The value .59 expresses expected depth as a fraction of the cap.

Suppose the sampled depth is two and a particular discrete action has probability .7 under π_2. The joint probability of that depth and action is $.48 \times .7 = .336$. The stored log-probability is $\log(.8) + \log(.6) + \log(.7)$, which records continuing, stopping and choosing the action. At the cap, the stopping factor is one and contributes zero log-probability. Although the environment receives only the action, the training record retains the depth that produced it.

A.4 Stopping probabilities and expected computation

Expected depth can be computed by adding the probabilities of reaching each refinement step. This gives a differentiable computation penalty without sampling a depth. For a fixed state, abbreviate $h_d = h_{\theta,d}(s)$ and $S_d = \prod_{j<d}(1 - h_j)$. Thus $S_1 = 1$ and S_d is the probability of reaching depth d. The cap is forced: only $h_1, \ldots, h_{d_{\max}-1}$ are free stopping probabilities.

Lemma 2 (Stopping-law identities)**.** *For any $h \in [0,1]^{d_{\max}-1}$, the distribution in Equation* (1) *is normalized, with*

$$\Pr(\tau \geq d \mid s) = S_d, \qquad \mathbb{E}[\tau \mid s] = \sum_{d=1}^{d_{\max}} S_d. \tag{6}$$

Its normalized mean $C(s) = \mathbb{E}[\tau \mid s]/d_{\max}$ satisfies, for $j < d_{\max}$,

$$\frac{\partial C(s)}{\partial h_j} = -\frac{S_j}{d_{\max}} \sum_{k=j+1}^{d_{\max}} \prod_{\ell=j+1}^{k-1} (1 - h_\ell) \leq 0. \tag{7}$$

Proof. Step 1: account for every possible final depth. Reaching depth d has probability S_d. Of that mass, $S_d h_d$ stops and $S_d(1 - h_d) = S_{d+1}$ continues. Thus, below the cap,

$$q(d) = S_d h_d = S_d - S_{d+1}, \qquad q(d_{\max}) = S_{d_{\max}}.$$

The last equality assigns all remaining probability to the forced stop. Adding these nonnegative masses cancels each intermediate survival term:

$$\sum_{d=1}^{d_{\max}} q(d) = (S_1 - S_2) + (S_2 - S_3) + \cdots + (S_{d_{\max}-1} - S_{d_{\max}}) + S_{d_{\max}} = S_1 = 1.$$

The same cancellation starting at d gives $\sum_{k=d}^{d_{\max}} q(k) = S_d$, which is exactly $\Pr(\tau \geq d \mid s)$. For $d_{\max} = 1$, the intermediate sums are empty and $q(1) = S_1 = 1$.

Step 2: count executed steps instead of final depths. For a realized stop at τ, the indicator $\mathbf{1}\{\tau \geq d\}$ is one for each executed refinement and zero for every unexecuted refinement. Consequently,

$$\tau = \sum_{d=1}^{d_{\max}} \mathbf{1}\{\tau \geq d\}, \qquad \mathbb{E}[\tau \mid s] = \sum_{d=1}^{d_{\max}} \Pr(\tau \geq d \mid s) = \sum_{d=1}^{d_{\max}} S_d.$$

This explains the two equivalent computations of 2.36 in Section A.3.

Step 3: differentiate the cost of the steps that can be avoided. Changing h_j cannot affect whether an earlier depth $k \leq j$ was reached. For a later depth $k > j$, factor its survival probability as

$$S_k = S_j(1 - h_j) \prod_{\ell=j+1}^{k-1} (1 - h_\ell).$$

Here S_j uses only earlier hazards and the product uses only later ones. Differentiating the finite sum from Step 2 therefore gives

$$\frac{\partial C}{\partial h_j} = \frac{1}{d_{\max}} \sum_{k=j+1}^{d_{\max}} \frac{\partial S_k}{\partial h_j} = -\frac{S_j}{d_{\max}} \sum_{k=j+1}^{d_{\max}} \prod_{\ell=j+1}^{k-1} (1 - h_\ell).$$

Every factor is nonnegative, so increasing a stopping probability cannot increase expected depth when the other hazards are held fixed. An empty product equals one, including the term $k = j + 1$. The expression is a polynomial, so it remains valid at $h_j = 1$ without dividing by $1 - h_j$. □

Equation (7) is a partial derivative in hazard coordinates; shared network parameters affect multiple hazards and representations, so the full parameter derivative uses the chain rule. For differentiable positive q_θ, the equivalent score identity is

$$\nabla_\theta C_\theta(s) = \mathbb{E}_{\tau \sim q_\theta} \left[\frac{\tau}{d_{\max}} \nabla_\theta \log q_\theta(\tau \mid s) \right]. \tag{8}$$

To obtain this identity, differentiate the finite expectation and use $\nabla_\theta q_\theta(d \mid s) = q_\theta(d \mid s) \nabla_\theta \log q_\theta(d \mid s)$:

$$\nabla_\theta C_\theta(s) = \sum_d \frac{d}{d_{\max}} \nabla_\theta q_\theta(d \mid s) = \sum_d q_\theta(d \mid s) \frac{d}{d_{\max}} \nabla_\theta \log q_\theta(d \mid s).$$

A state-only baseline can be subtracted while holding it fixed, since $\sum_d \nabla q_\theta(d \mid s) = 0$. The analytic expression in Equation (3) avoids sampling altogether. If a score estimator reuses old-policy depth samples, it needs importance weights for the current q_θ; differentiating a detached integer depth does not supply this gradient.

A.5 Returns, advantages and the shared value target

An observed return alone does not say whether a decision was good relative to what was already expected in that state. A state-only baseline $b_{\text{old}}(s_t)$ supplies that comparison. It is computed under the frozen rollout convention and does not condition on the sampled depth or action. An advantage is a signed estimate of the difference between an outcome and this baseline. A positive advantage favors the recorded joint decision; a negative advantage disfavors it. Each sampled outcome supplies credit to the recorded depth and action.

Generalized advantage estimation (GAE) combines successive temporal-difference residuals. A residual compares the current value prediction with one reward plus the discounted prediction at the next state:

$$\delta_t = r_t + \gamma m_t b_{\text{old}}(s_{t+1}) - b_{\text{old}}(s_t).$$

The mask m_t is zero after a true terminal transition, because no continuation value remains and one otherwise. For a finite rollout segment, the advantages can be accumulated backward as

$$\widehat{A}_t = \delta_t + \gamma \lambda_{\text{GAE}} m_t \widehat{A}_{t+1}, \qquad \widehat{A}_T = 0.$$

This recursion is the finite sum in Equation (12). The coefficient λ_{GAE} controls how much later residuals contribute; it is distinct from both loss weights λ_V and λ_c. At a nonterminal rollout cutoff, the last residual still includes the bootstrap value of the next state. Setting $\widehat{A}_T = 0$ ends the residual accumulation and does not remove that bootstrap.

For a worked calculation, take a two-transition segment with rewards .1 and .7, with the second transition terminal. Let the baseline predictions at the two visited states be .2 and .3 and use $\gamma = .9$ and $\lambda_{\mathrm{GAE}} = .8$ for this calculation. Then

$$\delta_t = .1 + .9(.3) - .2 = .17, \qquad \delta_{t+1} = .7 - .3 = .4,$$
$$\widehat{A}_t = .17 + .9(.8)(.4) = .458, \quad V_t^{\mathrm{targ}} = .2 + .458 = .658.$$

Every intermediate critic at state s_t receives the same target .658. The target is a bootstrapped return estimate; it need not equal the Monte Carlo return, which here is $.1 + .9(.7) = .73$. The example also shows why the target is not the observed outcome of every possible stopping depth: only the sampled depth and its action were executed.

The objective uses a state-only baseline and frozen targets. During a PPO update, the old baseline, advantage and value target are constants with respect to the current parameters. Section A.7 specifies the terminal and rollout-cutoff conventions.

A.6 Joint credit under PPO

The sampled pair (τ_t, a_t) is treated as one policy decision. Its log-probability separates into a depth term and an action term:

$$\log p_\theta(a_t, \tau_t \mid s_t) = \log q_\theta(\tau_t \mid s_t) + \log \pi_{\theta,\tau_t}(a_t \mid s_t).$$

The depth term includes each decision to continue before stopping. Before clipping and regularization, the likelihood-ratio actor signal is proportional to this log-probability's gradient multiplied by the advantage. It can therefore train stopping from task outcomes without a supervised label for the correct depth. Using only the action likelihood would omit the direct score term for the recorded stopping path.

For a sampled stop at depth two below the cap, $\log q_2 = \log(1 - h_1) + \log h_2$. Holding the other outputs fixed and taking interior probabilities,

$$\frac{\partial \log q_2}{\partial h_1} = -\frac{1}{1 - h_1}, \qquad \frac{\partial \log q_2}{\partial h_2} = \frac{1}{h_2}.$$

A positive advantage thus favors continuing past depth one and stopping at depth two for this sampled path. A negative advantage reverses these score directions. These are derivatives with respect to the output probabilities; shared network parameters couple the outputs and the final update also includes clipping, value regression and the computation penalty.

PPO compares the current probability of the recorded pair with the probability under the policy that collected it. The ratio ρ_t in Equation (10) exceeds one when that pair has become more likely. The denominator is fixed throughout the update. For example, changing the old joint probability .336 to a current probability .420 gives $\rho = 1.25$. With positive advantage .5 and clipping parameter $\eta = .2$, the two candidate contributions are $1.25(.5) = .625$ and $1.2(.5) = .6$. PPO uses their minimum, .6, so the minimized actor loss receives $-.6$ from this sample.

Clipping limits the surrogate's incentive to keep increasing an already-favored sampled decision. For negative advantages it similarly limits the incentive to keep decreasing a disfavored decision beyond the lower ratio threshold. Shared parameters and the other loss terms can still change the ratios after an update; clipping acts on the actor surrogate. The clipped objective is written explicitly in Equation (9).

A.7 The PPO surrogate and shared target convention

A rollout records both the external action and the internal depth that produced it. Keeping the depth makes it possible to assign likelihood-ratio credit to stopping as well as acting. Freeze a

rollout policy θ_{old} and store $(s_t, \tau_t, a_t, r_t, s_{t+1})$ together with its joint log-probability. With fixed advantages $\widehat{A}_t$, the actor loss is

$$\mathcal{L}_{\text{PPO}}(\theta) = -\widehat{\mathbb{E}}_t\Big[\min\Big\{\rho_t(\theta)\widehat{A}_t, \text{clip}(\rho_t(\theta), 1-\eta, 1+\eta)\widehat{A}_t\Big\}\Big], \tag{9}$$

where $\eta > 0$ is the clipping parameter (default 0.2) and

$$\rho_t(\theta) = \frac{q_\theta(\tau_t \mid s_t)\pi_{\theta,\tau_t}(a_t \mid s_t)}{q_{\theta_{\text{old}}}(\tau_t \mid s_t)\pi_{\theta_{\text{old}},\tau_t}(a_t \mid s_t)}. \tag{10}$$

The denominator is the frozen rollout policy's joint probability, or density for continuous actions and is nonzero on stored samples. Removing its stopping factor would remove direct likelihood-ratio credit for the sampled stopping decision. Although the environment observes only the marginal action policy $\pi_\theta(a \mid s) = \sum_d q_\theta(d \mid s)\pi_{\theta,d}(a \mid s)$, the recorded depth supports the joint estimator.

The shared GAE target is defined using the frozen, state-only bootstrap baseline $b_{\text{old}}(s)$ used by the rollout implementation. Let $m_t = 0$ after a true terminal transition and $m_t = 1$ otherwise and let $\lambda_{\text{GAE}} \in [0, 1]$ (default 0.95) be distinct from the loss coefficients. For a rollout segment ending at index $T-1$, the convention is

$$\delta_t = r_t + \gamma m_t b_{\text{old}}(s_{t+1}) - b_{\text{old}}(s_t), \tag{11}$$

$$\widehat{A}_t = \sum_{\ell=0}^{T-t-1} (\gamma\lambda_{\text{GAE}})^\ell \left(\prod_{k=0}^{\ell-1} m_{t+k}\right)\delta_{t+\ell}, \qquad V_t^{\text{targ}} = b_{\text{old}}(s_t) + \widehat{A}_t. \tag{12}$$

A nonterminal rollout cutoff bootstraps from $b_{\text{old}}(s_T)$. Segments end at environment resets. If a time limit truncates a continuing task, bootstrap from its final pre-reset observation, but do not extend the advantage trace into the reset episode. A baseline that depends on the sampled τ_t is not automatically valid for the joint score estimator. Targets, old probabilities and advantages remain fixed during a PPO update and every depth critic receives the same V_t^{targ} in Equation (2).

Local cost surrogate versus trajectory cost. The expectation of $C_\theta(s_t)$ in Equation (2) uses frozen rollout states. It differentiates the conditional depth distribution at those states, with no derivative through their sampling distribution. In contrast, the full discounted objective

$$J_c(\theta) = \mathbb{E}_{p_\theta} \sum_{t \geq 0} \gamma^t \left(r_t - \lambda_c \frac{\tau_t}{d_{\max}}\right) \tag{13}$$

also changes future state visits and future compute. A policy-gradient treatment of this objective would use return/advantage information including those future computation costs. Reward-only GAE plus a conditional cost penalty does not in general give that exact gradient. The same units yield the statewise coefficient $c = \lambda_c/d_{\max}$, but PPO clipping, advantage scaling, value training and distribution shift prevent identifying the training loss with the ideal stopping optimum.

A.8 Loss terms

The full loss in Equation (2) combines an actor term, an auxiliary value term and a computation term. The actor term connects stopping and action selection to task outcomes. The value term supplies a prediction task at every representation. The computation term expresses a preference for spending fewer refinements when the reward objective does not justify them.

Value supervision. At each recorded state, the squared error $(\widehat{v}_{\theta,d} - V_t^{\text{targ}})^2$ compares a depth's prediction with the common target. Averaging over depths keeps the total value-loss weight from scaling automatically with the number of heads. Even if depth two supplied the executed action, the critics at depths one, three and later depths still receive this loss. Their gradients also pass through the shared representations and refinement block. This can change action quality indirectly, which is why the factorial experiment separates auxiliary supervision from adaptive allocation. Supervising every value head does not imply that every action head receives a direct actor update on that sample.

Computation penalty. The normalized expectation $C_\theta(s)$ lies between $1/d_{\max}$ and one, because at least one refinement is performed and the cap is forced. It is differentiable through the current stopping probabilities. For the four-depth calculation above, the penalty is $\lambda_c(.59)$; if $\lambda_c = .01$,

it contributes $.0059$ before averaging across states. The expectation includes all possible stopping depths at that state, even though only one depth was sampled for the trajectory. Increasing an early stopping probability reduces the chance of paying for later refinements. Lemma 2 gives the exact derivative and covers probabilities at the boundary.

A sampled integer depth is a recorded outcome, not a differentiable probability. Treating that integer as a constant and differentiating its numerical value supplies no gradient to the halting network. The analytic expectation avoids this problem. A separate score-function estimator is possible, as shown in Equation (8), but its sampling distribution and any importance weights must be handled consistently.

Value differences. The difference $\widehat{v}_d - \widehat{v}_{d-1}$ compares two predictions of the same target. It is not a target for the halting head and need not equal the return gain from forcing another refinement. Proposition 3 derives the prediction target and gives a one-state example. The halting head receives task credit from the joint actor loss and depth credit from the computation penalty.

A.9 Shared-target regression

The auxiliary loss teaches each representation to predict the rollout target. It does not replace that target with the outcome of forcing a particular depth. Here $W_d(s)$ denotes the expected return after choosing an action from π_d at state s and then following a fixed continuation policy $\bar{\pi}$. Its action value $Q^{\bar{\pi}}(s, a)$ includes the immediate reward and subsequent discounted rewards. The distinction follows directly from the population minimizer of squared error.

Proposition 3 (Shared targets do not identify forced-depth values)**.** *Fix a joint distribution of a state S and a square-integrable shared target Y and fix each representation $Z_d = z_d(S)$. The minimizer of $\mathbb{E}[(f_d(Z_d) - Y)^2]$ over square-integrable measurable predictions is $f_d^*(Z_d) = \mathbb{E}[Y \mid Z_d]$, almost surely. This conditional mean need not equal $W_d(S) = \mathbb{E}_{a\sim\pi_d}[Q^{\bar{\pi}}(S, a)]$.*

Proof. Step 1: separate predictable signal from residual error. Fix a depth and write $Z = Z_d$, $m(Z) = \mathbb{E}[Y \mid Z]$ and $f(Z) = f_d(Z_d)$. The conditional mean $m(Z)$ is the average target among samples with the same representation. Write the prediction error as

$$f(Z) - Y = \underbrace{f(Z) - m(Z)}_{\text{prediction error relative to the mean}} + \underbrace{m(Z) - Y}_{\text{residual around that mean}}.$$

By the definition of a conditional mean, $\mathbb{E}[m(Z) - Y \mid Z] = 0$. Because $f(Z) - m(Z)$ is determined by Z, the cross term in the squared error satisfies

$$\mathbb{E}[(f(Z) - m(Z))(m(Z) - Y)] = \mathbb{E}[(f(Z) - m(Z))\mathbb{E}[m(Z) - Y \mid Z]] = 0.$$

Step 2: identify the minimizer. Expand the square and use the zero cross term:

$$\mathbb{E}[(f_d(Z_d) - Y)^2] = \mathbb{E}[(f_d(Z_d) - m_d(Z_d))^2] + \mathbb{E}[(Y - m_d(Z_d))^2]. \tag{14}$$

The second term is fixed by the data distribution and representation. The first term is nonnegative and becomes zero exactly when $f_d(Z_d) = m_d(Z_d)$ almost surely. This proves the population minimizer. It concerns the target Y actually supplied to regression; the calculation has not changed that target into an outcome obtained by forcing depth d.

Step 3: show why the two value notions can disagree. Consider a one-state bandit that ends after one action. At depth one, π_1 always selects an action with reward zero. At depth two, π_2 always selects an action with reward one. During data collection, choose depth two with probability $p \in (0, 1)$ and give both critics the observed reward Y. Then

$$\Pr(Y = 1) = p, \qquad \Pr(Y = 0) = 1 - p, \qquad \mathbb{E}[Y] = p.$$

Both representations are constant, so conditioning on either representation leaves that mean unchanged. The optimal predictions are therefore $f_1^* = f_2^* = p$. By contrast, executing each action head gives $W_1 = 0$ and $W_2 = 1$. Hence

$$f_2^* - f_1^* = 0 \quad \text{while} \quad W_2 - W_1 = 1.$$

Even exact shared-target regression can therefore give value differences that do not equal forced-depth return differences. □

When every representation retains all state information, the optimal predictions equal $\mathbb{E}[Y \mid S]$ at every depth. Information loss can make these conditional means differ; finite model classes, shared parameters, bootstrapping and optimization introduce further differences. Estimating forced-depth returns requires targets for those interventions. The joint policy ratio supplies reward credit to the sampled action and stopping path.

A.10 Invariance to value outputs

The architecture separates a training pathway from an inference pathway. During training, the loss on $V_d(z_d)$ updates the critic and shared representation. The updated representation can change both $\pi_d(z_d)$ and $h_d(z_d)$. At inference, the frozen policy computes actions and stopping directly from the representation. It does not read the numerical critic outputs.

Proposition 4 (Invariance to value-output replacement)**.** *Fix the encoder, refinement block, action heads and halting heads. Replace each auxiliary value output by an arbitrary function $\widetilde{V}_d(z_d)$ without modifying any of those fixed mappings or the stopping algorithm. Then the joint action/depth law $p_\theta(a, d \mid s)$ in Equation* (1)*, the deterministic threshold stopping depth and the expected environment return are unchanged.*

Proof. Step 1: trace one forward decision. The input and encoder are fixed, so z_0 is unchanged. Inductively, an unchanged z_{d-1} and refinement block produce the same z_d. The fixed action and halting heads then produce the same π_d and h_d at every depth. Replacing only $V_d(z_d)$ affects none of these mappings.

Step 2: compare the two stopping conventions. Under stochastic stopping, the products defining S_d and $q(d \mid s)$ use only the unchanged hazards. Thus both $q(d \mid s)$ and $q(d \mid s)\pi_d(a \mid s)$ are unchanged. Under thresholded evaluation, the first depth with $h_d > 1/2$, or the forced cap, is also unchanged. Its selected action distribution is the same.

Step 3: extend the equality to trajectories. Begin with the same initial-state distribution and environment dynamics. At the first state, identical action kernels give the same distribution of the next state and reward. Repeating this argument at each step gives identical finite trajectory distributions. Their expected returns are therefore equal whenever those returns are defined, including bounded-horizon and bounded-reward discounted settings. □

This result concerns a frozen forward policy. Changing value targets during training can change the shared representation and is therefore a different intervention. Replacing the stopping rule by a threshold on value increments would also define a different evaluation policy.

A.11 Training loop

A rollout is a sequence of environment interactions collected before updating the policy. For one update cycle, the parameters used to collect that data are denoted by θ_{old} and kept fixed as the reference policy. For each state, the training procedure computes the full refinement chain, samples a depth from $q_{\theta_{\text{old}}}$ and samples the action from the selected head. Only that action is executed. The other representations remain available for auxiliary value supervision; computing them does not create additional environment transitions.

The stored record contains the state, selected depth, executed action, reward, next state, termination information and old joint log-probability. The rollout baseline and subsequent rewards then determine the fixed advantages and value targets. Old joint probabilities and targets must remain associated with the same recorded transition when samples are batched or shuffled. Otherwise the likelihood ratio or regression target would describe a different decision.

For a batch of N recorded states, the stopping distribution can be viewed as an array with N rows and $d_{\max}$ columns: each row is a normalized distribution over depths. The value predictions form another array of the same shape, but all entries in a row are compared with one shared target for that state. The actor likelihood selects the recorded depth's action head for each row. The depth penalty sums $d\, q_\theta(d \mid s)/d_{\max}$ along that row.

Each optimization pass recomputes the current representations, probabilities and value predictions on the recorded states. The three losses are averaged and combined as in Equation (2) and differ-

entiation updates the trainable parameters. The old-policy quantities remain fixed while the current quantities change. After the update cycle, new trajectories are collected with the updated policy. Algorithm 1 summarizes these operations; Section A.7 distinguishes this rollout-state cost surrogate from a full trajectory objective that includes future computation costs.

Algorithm 1 AIC: full-depth training and adaptive-depth evaluation.

Inputs: encoder, shared refinement block, depth-specific heads, $d_{\max}$, λ_c, λ_V.

1. Freeze rollout parameters θ_{old}. For each state, compute z_0 and all z_d, π_d, V_d; compute h_d for $d < d_{\max}$.
2. Sample $\tau \sim q_{\theta_{\text{old}}}(\cdot \mid s)$, with a forced stop at $d_{\max}$ and sample $a \sim \pi_{\theta_{\text{old}},\tau}$.
3. Execute a; store the transition, τ and $\log q_{\theta_{\text{old}}}(\tau \mid s) + \log \pi_{\theta_{\text{old}},\tau}(a \mid s)$.
4. Construct frozen rollout advantages and a shared value target at every state as in Equation (12).
5. For each PPO update, recompute all depths and the joint ratio in Equation (10). Minimize Equation (2), differentiating the expected normalized depth through q_θ.
6. At evaluation, refine sequentially and stop at the first $d < d_{\max}$ with $h_d > 0.5$, or at the cap; execute an action from $\pi_{\theta,d}$.

A.12 Deployment and computation accounting

Deployment uses the learned network to process one state sequentially. It encodes the input, applies the shared refinement block and stops at the first depth below the cap for which $h_d > 0.5$. If no such depth occurs, it stops at $d_{\max}$. The selected action head then supplies the action. Later refinements need not be evaluated to select that action, because all earlier stopping decisions depend only on representations already computed.

In the four-depth example, $h_1 = .2$ continues and $h_2 = .6$ stops, giving evaluation depth two compared with stochastic mean depth 2.36. The action is drawn from the selected action distribution.

Thresholding is a different evaluation policy. Define $\tau_{\text{eval}}(s) = \min(\{d < d_{\max} : h_d(s) > 1/2\} \cup \{d_{\max}\})$. This is neither the mean nor necessarily the mode of q_θ. For example, $d_{\max} = 3$ and $h_1 = h_2 = 0.4$ give $q = (0.4, 0.24, 0.36)$, with mean depth 1.96, modal depth one and thresholded depth three. Near-binary hazards can reduce the discrepancy. We measure inference computation and return under the deployed threshold policy.

The expected-depth penalty describes refinement usage, not all execution costs. Encoding, output heads, memory movement and control flow also take time. An expensive frozen encoder can dominate runtime even when many refinements are skipped and batching can affect whether different stopping depths translate into latency savings. Training computes all depths for supervision, so an early sampled stopping depth does not imply the same saving during training. These distinctions motivate the separate measurements of depth, inference latency and training time in Section Q.

Finally, the allocation theory studies what could be gained if depth were assigned using known depth-specific action values under its assumptions. Those values, denoted by W_d, are expected returns from selecting a particular depth's action and then following a fixed continuation policy. They differ from the shared-target predictions $\widehat{v}_d$. AIC learns a stopping policy from sampled outcomes and the experiments evaluate the resulting reward and computation.

B Theory of State-Dependent Computation Allocation

We study whether assigning computation to the states that need it can improve value without increasing average depth. We answer this question for a fixed state distribution and a fixed family of depth-specific action policies. The comparison changes only how depth is assigned.

B.1 Setup, units and assumptions

Let μ be a fixed probability distribution over states and let the integer depth cap satisfy $d_{\max} \geq K \geq 2$. A stopping law $q(d \mid s)$ is a probability distribution on $\{1, \ldots, d_{\max}\}$. For integrable decision

values $W_d(s)$ and an integrable reference $W^*(s) \geq W_d(s)$, define

$$\mathcal{B}_\mu(q) := \mathbb{E}_{s\sim\mu} \sum_{d=1}^{d_{\max}} d\, q(d \mid s), \tag{15}$$

$$\mathcal{R}_\mu(q) := \mathbb{E}_{s\sim\mu} \sum_{d=1}^{d_{\max}} q(d \mid s)\big[W^*(s) - W_d(s)\big]. \tag{16}$$

The budget $\mathcal{B}_\mu$ counts expected refinement steps under μ. The training penalty instead averages the statewise normalized cost $C_\theta(s) = \mathbb{E}_\theta[\tau \mid s]/d_{\max}$ over rollout states. The mean-depth budget b used below is therefore distinct from the nominal experimental cap B. The expectation over $s \sim \mu$ averages over states, while the sum over d averages over the stopping choice at a given state. Thus $\mathcal{B}_\mu$ is average work and $\mathcal{R}_\mu$ is average value lost relative to the reference. All allocations use the same fixed action-policy family and continuation policy, with no retraining. Here W_d is a decision value used for analysis, not an auxiliary prediction $\widehat{v}_d$. The following assumptions describe when easy states can stop early while hard states need additional depth.

(A1) A measurable difficulty partition. The state space is partitioned into easy states E and hard states H, with $\mu(H) = \epsilon \in (0, 1)$ and $\mu(E) = 1 - \epsilon$.

(A2) Attainability. For $s \in E$, $W_d(s) = W^*(s)$ at every depth. For $s \in H$, $W_d(s) = W^*(s)$ at every $d \geq K$.

(A3) Shortfall before sufficient depth. There is a scale $\Delta > 0$ such that for every $s \in H$ and $1 \leq d < K$,

$$W^*(s) - W_d(s) \geq \Delta(1 - d/K). \tag{17}$$

All statements hold μ-almost surely. The scale Δ controls shortfall before sufficient depth; it is not a separately assumed action gap. Assumptions (A2)–(A3) make K the first optimal depth on a hard state and the ideal allocator knows the easy/hard partition. The allocation result is conditional on these decision-value assumptions.

B.2 A tight worked example

The benefit comes from associating depth with the state. The following construction makes this association concrete and will attain the theorem's bound exactly. Consider four equally likely states, which we call contexts in this one-decision example. Three have $W_d = 1$ at every depth; the fourth has $W_d = d/4$ for $d = 1, \ldots, 4$. Set $K = d_{\max} = 4$, $\epsilon = 1/4$ and $\Delta = 1$. The ideal allocator uses depths $(1, 1, 1, 4)$, so its mean value is one and its mean depth is $b = 7/4$.

The adaptive rule spends one step on each of the three easy contexts and four on the hard context. A state-independent rule cannot direct its deeper choices toward the hard context: it uses the same depth distribution in all four contexts.

A feasible deterministic constant-depth baseline uses depth one, with mean value $(3 + 1/4)/4 = 13/16 = 0.8125$. Every state-independent schedule with mean depth $7/4$ has mean value

$$\frac{3}{4} + \frac{1}{4}\,\mathbb{E}[\tau/4] = \frac{3}{4} + \frac{7}{64} = \frac{55}{64} = 0.859375. \tag{18}$$

For example, choose depths one and two with probabilities $1/4$ and $3/4$. Its shortfall is $9/64 = 0.140625$, exactly the bound $\epsilon\Delta(1 - b/K)$. This finite-context construction attains the lower bound in Equation (5).

B.3 Proof of Theorem 1, including randomized schedules

The proof has two parts: construct an allocation with zero shortfall, then show that every state-independent schedule at its mean-depth budget has positive shortfall. Randomization is allowed in the baseline, so the argument does not rely on comparing a fractional mean depth with an integer fixed depth.

Proof. Step 1: construct the state-dependent allocation. Let q^* stop at depth one on the easy set E and at depth K on the hard set H. Assumption (A2) makes both choices attain $W^*(s)$, so every

statewise shortfall is zero and $\mathcal{R}_\mu(q^*) = 0$. Its expected work is

$$\mathcal{B}_\mu(q^*) = (1-\epsilon)\cdot 1 + \epsilon K = b := 1 + \epsilon(K-1).$$

The easy states make this budget smaller than K: $K - b = (1-\epsilon)(K-1) > 0$.

Step 2: lower-bound shortfall for any chosen depth. Write $(x)_+ = \max\{x, 0\}$. For a hard state and $d < K$, (A3) gives shortfall at least $\Delta(1 - d/K)$. For $d \geq K$, (A2) gives zero shortfall and $(1 - d/K)_+ = 0$. Both cases are therefore captured by

$$W^*(s) - W_d(s) \geq \Delta(1 - d/K)_+ \qquad (s \in H).$$

Easy states have zero shortfall.

Step 3: average under a state-independent schedule. Let $\tau \sim \nu$ have $\mathbb{E}_\nu[\tau] \leq b$ and use this same distribution at every state. The depth is independent of whether the sampled state is hard. Since hard states have probability ϵ, Step 2 gives

$$\mathcal{R}_\mu(\nu) \geq \mathbb{E}_{s\sim\mu,\,\tau\sim\nu}[\mathbf{1}\{s \in H\}\Delta(1 - \tau/K)_+] = \epsilon\Delta\,\mathbb{E}_{\tau\sim\nu}[(1 - \tau/K)_+].$$

Independence is essential to this factorization; it need not hold for a state-dependent rule.

Step 4: use only the baseline's average budget. The function $f(x) = (1 - x/K)_+$ is the maximum of two affine functions, $1 - x/K$ and zero and is therefore convex. Jensen's inequality gives

$$\begin{aligned}\mathcal{R}_\mu(\nu) &\geq \epsilon\Delta\,\mathbb{E}[f(\tau)] \geq \epsilon\Delta\, f(\mathbb{E}[\tau]) \\ &\geq \epsilon\Delta\, f(b) = \epsilon\Delta(1 - b/K) > 0. \end{aligned} \tag{19}$$

The third inequality uses that f is non-increasing and $\mathbb{E}[\tau] \leq b$. The last step uses $b < K$ from Step 1. This proof also allows baseline mass above K: the positive-part function already includes those depths. □

Deterministic fixed-depth special case. For a constant integer depth $K' \leq b$, feasibility requires $K' \leq \lfloor b \rfloor < K$. Directly applying (A3) gives the sharper depth-specific bound

$$\mathcal{R}_\mu(K') \geq \epsilon\Delta(1 - K'/K). \tag{20}$$

The comparison uses an upper compute budget; an integer baseline need not consume exactly b. A distribution over $\lfloor b \rfloor$ and $\lceil b \rceil$ can consume b exactly and Equation (19) still applies.

B.4 More than two difficulty levels and the saturated case

The same argument can be applied separately to groups that need different depths. The resulting lower bound adds the contribution from each group.

Proposition 5 (Multiple difficulty levels)**.** *Suppose the hard set is partitioned into finitely many levels H_i with conditional masses $p_i > 0$, $\sum_i p_i = 1$, earliest optimal depths $2 \leq K_i \leq d_{\max}$ and shortfall scales $\Delta_i > 0$. Replace (A2)–(A3) on H_i by their versions with K_i, Δ_i. The ideal state-dependent allocator has zero shortfall and mean depth*

$$b^* = 1 + \epsilon \sum_i p_i(K_i - 1). \tag{21}$$

Every state-independent schedule with $\mathbb{E}[\tau] \leq b^*$ obeys

$$\mathcal{R}_\mu \geq \epsilon \sum_i p_i \Delta_i (1 - b^*/K_i)_+. \tag{22}$$

A constant depth K' instead obeys the bound with b^* replaced by K'.

Proof. Step 1: construct the allocation and compute its budget. Use depth one on E and depth K_i on H_i. Each group then attains its reference value, giving zero shortfall. Because $\mu(H_i) = \epsilon p_i$, the mean depth is

$$(1-\epsilon)\cdot 1 + \sum_i \epsilon p_i K_i = 1 + \epsilon \sum_i p_i(K_i - 1) = b^*.$$

Step 2: bound each group's contribution. For a state-independent depth τ, the shortfall on H_i is at least $\Delta_i(1-\tau/K_i)_+$. Independence permits averaging over depths before weighting by the group's probability:

$$\mathcal{R}_\mu \geq \epsilon \sum_i p_i \Delta_i \mathbb{E}[(1-\tau/K_i)_+].$$

For each i, convexity and the budget condition imply

$$\mathbb{E}[(1-\tau/K_i)_+] \geq (1-\mathbb{E}[\tau]/K_i)_+ \geq (1-b^*/K_i)_+.$$

Substituting this inequality into the sum proves Equation (22). For a constant depth K', substitute $\tau = K'$ directly instead. □

The bound can be zero for a group with $K_i \leq b^*$; no positive separation is claimed from that group alone. At a fixed baseline depth, its summand increases with K_i. Changing K_i also changes the ideal budget b^*, so this observation does not imply unrestricted monotonicity when both are varied.

At the other extreme, if all depths produce the same decision value, there is no reward benefit to offset extra work.

Corollary 6 (No benefit from depth)**.** *If $W_d(s) = W_1(s)$ at every state and depth, a strictly positive cost c per refinement step makes depth one the unique optimal stopping depth in the ideal statewise problem.*

Proof. For every $d > 1$, $[W_d(s) - cd] - [W_1(s) - c] = -c(d-1) < 0$. A distribution placing positive mass on such a depth has lower utility than stopping at one. This gives a unique stopping distribution, although hazards after the certain stop are arbitrary. □

DistShift's saturated result is compatible with this condition; the statement does not guarantee that training finds this optimum.

B.5 Sequential returns

The fixed-μ analysis compares decisions on a common state distribution. In a sequential MDP, changing the stopping policy can also change which states are visited. The next identity makes the required state distribution and continuation values explicit before relating shortfall to total return.

Lemma 7 (Discounted occupancy identity)**.** *Consider a discounted MDP with bounded rewards, discount $0 \leq \gamma < 1$ and a fixed initial-state distribution. Let π^* be an optimal policy and let $\pi_q(a \mid s) = \sum_d q(d \mid s)\pi_d(a \mid s)$. Define the normalized discounted occupancy*

$$d_{\pi_q}(s) = (1-\gamma)\sum_{t\geq 0} \gamma^t \Pr_{\pi_q}(s_t = s),$$

with the analogous measure definition for non-discrete states. If $W^* = V^*$ and $W_d(s) = \mathbb{E}_{a\sim\pi_d}[Q^*(s,a)]$, then

$$J(\pi^*) - J(\pi_q) = \frac{\mathcal{R}_{d_{\pi_q}}(q)}{1-\gamma}. \tag{23}$$

Proof. Step 1: express shortfall as a one-transition residual. Define the statewise shortfall $\ell_q(s) = V^*(s) - \sum_d q(d \mid s) W_d(s)$. The Bellman definition of Q^* gives

$$\mathbb{E}[r_t + \gamma V^*(s_{t+1}) \mid s_t = s] = \mathbb{E}_{a\sim\pi_q}[Q^*(s,a)] = \sum_d q(d \mid s) W_d(s).$$

Therefore $\ell_q(s_t) = \mathbb{E}[V^*(s_t) - r_t - \gamma V^*(s_{t+1}) \mid s_t]$.

Step 2: add the residuals along a trajectory. Multiply by γ^t, sum through T and take expectation under π_q:

$$\begin{aligned}\mathbb{E}_{\pi_q}\sum_{t=0}^{T}\gamma^t \ell_q(s_t) &= \mathbb{E}_{\pi_q}\sum_{t=0}^{T}[\gamma^t V^*(s_t) - \gamma^t r_t - \gamma^{t+1}V^*(s_{t+1})] \\ &= \mathbb{E}[V^*(s_0)] - \mathbb{E}_{\pi_q}\sum_{t=0}^{T}\gamma^t r_t - \gamma^{T+1}\mathbb{E}_{\pi_q}[V^*(s_{T+1})].\end{aligned}$$

The value terms at $s_1, \ldots, s_T$ cancel in pairs. If rewards are bounded in magnitude by $R_{\max}$, then $|V^*(s)| \le R_{\max}/(1-\gamma)$. The remaining terminal term is bounded by $\gamma^{T+1} R_{\max}/(1-\gamma)$ and tends to zero. The limit is $J(\pi^*) - J(\pi_q)$, since the initial-state distribution is the same for both policies.

Step 3: rewrite the trajectory sum as a state average. The factor $(1-\gamma)$ normalizes the discounted occupancy because $(1-\gamma)\sum_{t\ge 0}\gamma^t = 1$. By its definition,

$$\mathbb{E}_{s\sim d_{\pi_q}}[\ell_q(s)] = (1-\gamma)\mathbb{E}_{\pi_q}\sum_{t\ge 0}\gamma^t \ell_q(s_t) = \mathcal{R}_{d_{\pi_q}}(q).$$

Combining this equality with Step 2 and dividing by $1-\gamma$ proves Equation (23). □

The same occupancy enters the deployed compute budget: $(1-\gamma)\mathbb{E}_{\pi_q}\sum_{t\ge 0}\gamma^t \tau_t = \mathcal{B}_{d_{\pi_q}}(q)$. Because two policies can have different occupancies, equal computation under an arbitrary μ need not mean equal computation along their trajectories. Applying the theorem to sequential returns therefore requires the value-profile assumptions and hard-state mass under the relevant occupancies, together with a compute check for each policy. The identity also uses Q^*: a general common continuation value $Q^{\bar\pi}$ cannot replace it without further assumptions.

C Optimal Stopping for a Fixed Value Profile

If the return available at each depth were known at one state, when should computation stop? The analysis concerns fixed decision values; AIC learns a halting policy from sampled outcomes rather than receiving this profile as input.

C.1 A fixed-policy, fixed-state subproblem

Fix a state s, all depth-specific action distributions π_d and a common continuation policy $\bar\pi$ at subsequent environment steps. Define

$$W_d(s) = \mathbb{E}_{a\sim\pi_d(\cdot|s)}[Q^{\bar\pi}(s,a)], \qquad U_d(s;c) = W_d(s) - cd, \quad c \ge 0. \tag{24}$$

The quantity W_d changes the current action distribution while holding future behavior fixed. No environment transition or additional temporal discount occurs during a refinement step. The profile is treated as fixed and available to the ideal decision maker. It is distinct from the learned shared-target predictions in Equation (4). At the same unscaled reward units, the normalized depth penalty corresponds to $c = \lambda_c/d_{\max}$.

C.2 Constant cost and adaptive depth

State-dependent stopping does not require a state-dependent price for computation. With fixed values $W_d(s)$ and a constant price $c > 0$, the ideal decision is $\max_d[W_d(s) - cd]$. Its maximizing depth can vary because the benefit of refinement varies across states. This also applies to ACT-style ponder objectives: a flat cost is compatible with adaptive computation.

In the four-context construction of Section B, every additional step on an easy context has value increment zero, whereas every additional step on the hard context has value increment $1/4$. For $0 < c < 1/4$, utility strictly decreases with depth on the easy contexts and strictly increases on the hard one. The unique optimal depths are therefore one and four, respectively, giving precisely the ideal allocation. At $c = 1/4$ all hard-context depths tie; for $c > 1/4$ depth one is optimal everywhere. Thus the strict interval on c is necessary for that example's unique adaptive solution.

The learning problem is to infer the useful depth from information available before stopping. AIC addresses that problem through joint RL training and tests when the resulting allocation explains performance gains.

C.3 The general optimal continuation condition

At a given depth, continuing is useful when the best later outcome compensates for the remaining computation cost. The next recursion compares stopping now with that full continuation opportunity.

Lemma 8 (Finite-depth stopping recursion)**.** *After reaching depth d, treat its computation as sunk and define*

$$F_d(s) = \max_{d \le k \le d_{\max}} \{W_k(s) - c(k-d)\}. \tag{25}$$

Then $F_{d_{\max}} = W_{d_{\max}}$ and, for $d < d_{\max}$,

$$F_d(s) = \max\{W_d(s),\, F_{d+1}(s) - c\}. \tag{26}$$

Stopping at d is optimal among remaining depths exactly when $W_d(s) \ge F_{d+1}(s) - c$; equality permits either stopping or an optimal continuation.

Proof. Step 1: handle the final depth. At $d = d_{\max}$, there is no remaining refinement. The only feasible choice in Equation (25) is $k = d_{\max}$, so $F_{d_{\max}} = W_{d_{\max}}$.

Step 2: split the remaining choices into stop and continue. At $d < d_{\max}$, choosing $k = d$ yields W_d with no additional computation cost. For every later choice $k \ge d+1$, separate the cost of the next step:

$$W_k - c(k-d) = \big[W_k - c(k-(d+1))\big] - c.$$

The maximum of the bracket over $k \ge d+1$ is exactly F_{d+1}. Taking the larger of the stop and continue options gives $F_d = \max\{W_d, F_{d+1} - c\}$. Stopping is therefore optimal exactly when $W_d \ge F_{d+1} - c$. Equality means that both options attain the same value.

Step 3: reconcile remaining cost with total cost. At depth d, the first d steps have already been paid for. Subtracting their constant cost cd gives

$$F_d - cd = \max_{k \ge d}\{W_k - c(k-d) - cd\} = \max_{k \ge d}(W_k - ck).$$

In particular, $\max_k U_k = F_1 - c$. The first refinement costs c too; it does not affect which depth maximizes utility. □

Without additional structure, the relevant benefit is the best continuation value, not just $W_{d+1} - W_d$. For stochastic internal transitions, continuation would require a conditional expectation over future internal states.

C.4 First-crossing rule

A one-step threshold is sufficient when successive benefits diminish. Once an increment falls below cost, no later increment can make continued refinement worthwhile under this assumption.

Proposition 9 (Diminishing returns is a sufficient condition)**.** *Let $g_d(s) = W_{d+1}(s) - W_d(s)$ for $d < d_{\max}$. If $g_d(s)$ is non-increasing in d, the earliest maximizing depth is*

$$d^*(s) = \min\big(\{d < d_{\max} : g_d(s) \le c\} \cup \{d_{\max}\}\big). \tag{27}$$

Proof. Step 1: translate value increments into utility increments. Since $U_d = W_d - cd$,

$$U_{d+1} - U_d = (W_{d+1} - W_d) - c = g_d - c.$$

The assumption that g_d is non-increasing also makes these utility increments non-increasing.

Step 2: compare the first crossing with every earlier depth. Suppose the threshold set is nonempty and let d_0 be its first element. Before d_0, each $g_j - c$ is strictly positive. For any $k < d_0$, adding those increments gives

$$U_{d_0} - U_k = \sum_{j=k}^{d_0-1} (g_j - c) > 0.$$

Thus no earlier depth is optimal.

Step 3: compare it with every later depth. At the crossing, $g_{d_0} \le c$. Non-increasing increments imply $g_j \le c$ for every $j \ge d_0$. Hence for any $k > d_0$,

$$U_k - U_{d_0} = \sum_{j=d_0}^{k-1} (g_j - c) \le 0.$$

No later depth has greater utility, although zero increments can create ties. This proves that d_0 is the earliest global maximizer.

Step 4: cover the cap and single-depth cases. If the threshold set is empty, every utility increment is positive and the cap is the unique maximizer. If $d_{\max} = 1$, there are no increments and depth one is the only feasible choice. The stated formula covers both cases. □

With $\Delta W_{d+1} := W_{d+1} - W_d$, the familiar condition

$$\Delta W_{d+1}(s) \le c \tag{28}$$

is therefore sufficient under diminishing returns. It is not a general stopping condition for arbitrary profiles and evaluating its next-depth counterfactual can itself require computation.

Counterexample without diminishing returns. Let $(W_1, W_2, W_3) = (0, 0, 3)$ and $c = 1$. The first increment is zero, so a first-crossing heuristic stops at one. The utilities are $(U_1, U_2, U_3) = (-1, -2, 0)$ and depth three is uniquely optimal. The recursion continues because the later benefit compensates for both extra steps. This is why an observed low immediate increment does not by itself justify a learned early exit.

C.5 Optimal stopping distributions and hazard gradients

Randomized stopping forms a weighted average of the fixed depth utilities. Its optimum therefore depends on which depths maximize utility, as the following result makes precise.

Theorem 10 (Existence and support of an optimal stopping distribution)**.** *For fixed $W_d(s)$ and c, maximize*

$$J_s(h) = \sum_{d=1}^{d_{\max}} q_h(d \mid s) U_d(s; c) \tag{29}$$

over unrestricted hazards $h \in [0,1]^{d_{\max}-1}$. The maximum is $M = \max_d U_d$. A deterministic distribution at any maximizing depth attains it and a stopping distribution is optimal if and only if it is supported on maximizing depths. Under Proposition 9's assumption, its first-crossing depth is one such choice.

Proof. Step 1: bound every randomized choice by the best depth. The stopping-law lemma gives $q_h(d \mid s) \ge 0$ and $\sum_d q_h(d \mid s) = 1$. Since $M = \max_d U_d$, each difference $M - U_d$ is nonnegative and

$$M - J_s(h) = M \sum_d q_h(d \mid s) - \sum_d q_h(d \mid s) U_d = \sum_d q_h(d \mid s)(M - U_d) \ge 0.$$

Equality holds exactly when every depth given positive probability has $U_d = M$. This proves the support condition: randomizing over suboptimal depths cannot attain the optimum.

Step 2: show that the bound is attainable with hazards. Choose a maximizing depth d_0. Set all earlier hazards to zero, so the policy certainly reaches it. If $d_0 < d_{\max}$, set $h_{d_0} = 1$; if $d_0 = d_{\max}$, the cap supplies the forced stop. This realizes $q_h(d_0 \mid s) = 1$ and attains M. Any hazards after a certain stop have no effect.

Step 3: verify that mixtures over tied optima are feasible too. More generally, let ν be any distribution over depths and define its remaining mass $R_d = \sum_{k=d}^{d_{\max}} \nu(k)$. For $d < d_{\max}$ with $R_d > 0$, choose $h_d = \nu(d)/R_d$. Start with $S_1 = R_1 = 1$. If $S_d = R_d > 0$, then

$$q_h(d) = S_d h_d = \nu(d), \qquad S_{d+1} = S_d(1 - h_d) = R_d - \nu(d) = R_{d+1}.$$

If $R_d = 0$, all later masses are zero and their hazards can be arbitrary. Induction gives the desired probabilities below the cap and $q_h(d_{\max}) = R_{d_{\max}} = \nu(d_{\max})$ at the cap. Every distribution supported on maximizing depths is therefore realizable and optimal. □

A unique maximizing depth gives a unique optimal depth distribution, while hazards after a certain stop remain arbitrary. Sigmoid heads with finite logits lie in $(0, 1)$, so deterministic boundary distributions may only be approached. Shared network parameters also couple decisions across states

and independently optimal depths need not be jointly realizable. The optimization is over free hazards for a fixed value profile; the jointly parameterized PPO training problem is analyzed separately through its surrogate.

For a non-optimal stopping distribution, the local hazard gradient compares stopping now with the continuation strategy currently in use.

Proposition 11 (The local hazard compares stopping with continuation)**.** *For $d < d_{\max}$, let the downstream stopping law after forcing continuation past d be*

$$\widetilde{q}_{d,k} = h_k \prod_{j=d+1}^{k-1} (1 - h_j) \; (d < k < d_{\max}), \qquad \widetilde{q}_{d,d_{\max}} = \prod_{j=d+1}^{d_{\max}-1} (1 - h_j),$$

and define $\overline{U}_{d+} = \sum_{k=d+1}^{d_{\max}} \widetilde{q}_{d,k} U_k$. Holding the value profile and all other hazards fixed,

$$\frac{\partial J_s}{\partial h_d} = S_d(U_d - \overline{U}_{d+}). \tag{30}$$

Proof. Step 1: isolate decisions affected by h_d. The total utility from stops before d is $A_{<d} = \sum_{k<d} q_h(k \mid s) U_k$ and does not depend on h_d. Reaching d has probability S_d, which also depends only on earlier hazards. Once d is reached, the policy stops with probability h_d or continues with probability $1 - h_d$.

Step 2: write the value of continuing. After forcing continuation, the downstream hazards define $\widetilde{q}_{d,k}$ on depths $k > d$. The stopping-law telescoping calculation shows that these probabilities sum to one. Their weighted utility is $\overline{U}_{d+}$. It uses only later hazards and remains well defined even if $S_d = 0$. Consequently,

$$J_s(h) = A_{<d} + S_d\big[h_d U_d + (1 - h_d)\overline{U}_{d+}\big].$$

Step 3: differentiate this expression. Holding the other hazards and all U_k fixed makes $A_{<d}$, S_d and $\overline{U}_{d+}$ constant with respect to h_d. Thus

$$\frac{\partial J_s}{\partial h_d} = S_d \frac{\partial}{\partial h_d}[h_d U_d + (1 - h_d)\overline{U}_{d+}] = S_d(U_d - \overline{U}_{d+}).$$

This proves the formula, including boundary hazards as a polynomial derivative. □

Thus the gradient compares current utility with the current downstream strategy, which may differ from both the next depth and the optimal continuation. If $S_d = 0$, changing the hazard cannot affect the objective because that depth is never reached. For $h_d = \sigma(\ell_d)$, the logit derivative also contains $h_d(1 - h_d)$ and can be small near saturation.

C.6 Compute prices and value-estimation error

We first vary the cost while holding the value profile fixed, then bound the effect of uniformly inaccurate values.

Proposition 12 (Optimal depth is non-increasing in its price)**.** *Fix the value profile. If $0 \le c_1 < c_2$ and $d_i \in \arg\max_d [W_d - c_i d]$, then $d_2 \le d_1$, without a diminishing-returns assumption.*

Proof. Compare the two chosen depths under each price. Because d_1 is optimal at c_1 and d_2 is optimal at c_2,

$$W_{d_1} - c_1 d_1 \ge W_{d_2} - c_1 d_2,$$
$$W_{d_2} - c_2 d_2 \ge W_{d_1} - c_2 d_1.$$

Adding cancels both value terms and leaves

$$(c_2 - c_1)(d_1 - d_2) \ge 0.$$

The price difference is strictly positive, so division gives $d_1 - d_2 \ge 0$. The argument permits ties in either maximization and does not require monotone value increments. □

This comparative statement holds with the action/value profile frozen; retraining at a different λ_c can change that profile and need not preserve the relation.

Proposition 13 (A uniform-error guarantee for value-based selection)**.** *Suppose a value estimate $\widetilde{W}_d$ satisfies $\max_d |\widetilde{W}_d - W_d| \le \xi$. Let $d^* \in \arg\max_d(W_d - cd)$ and $\widetilde{d} \in \arg\max_d(\widetilde{W}_d - cd)$. Then*

$$0 \le U_{d^*} - U_{\widetilde{d}} \le 2\xi. \tag{31}$$

If the true maximizer is unique with gap greater than 2ξ to every other depth, the estimated maximizer equals it.

Proof. Step 1: transfer the value error to utility. Both utilities subtract the same exact cost cd. Therefore, for $\widetilde{U}_d = \widetilde{W}_d - cd$, $|\widetilde{U}_d - U_d| = |\widetilde{W}_d - W_d| \le \xi$.

Step 2: insert the estimated utilities into the regret. Add and subtract the estimates at the two selected depths:

$$\begin{aligned} U_{d^*} - U_{\widetilde{d}} &= (U_{d^*} - \widetilde{U}_{d^*}) + (\widetilde{U}_{d^*} - \widetilde{U}_{\widetilde{d}}) + (\widetilde{U}_{\widetilde{d}} - U_{\widetilde{d}}) \\ &\le \xi + 0 + \xi = 2\xi. \end{aligned}$$

The middle term is nonpositive because $\widetilde{d}$ maximizes the estimated utility. The outer terms are each at most ξ by Step 1. True optimality of d^* gives the lower bound zero.

Step 3: use a strict gap to obtain exact selection. If every other depth loses more than 2ξ in true utility, selecting such a depth would violate Step 2. Thus a unique optimum with that gap must also be the estimated optimum. □

The corresponding bound for an adjacent increment follows from the triangle inequality:

$$\left|(\widetilde{W}_{d+1} - \widetilde{W}_d) - (W_{d+1} - W_d)\right| \le |\widetilde{W}_{d+1} - W_{d+1}| + |\widetilde{W}_d - W_d| \le 2\xi.$$

If the true first-crossing rule is valid and every examined increment has margin $|g_d - c| > 2\xi$, its threshold decisions are preserved. This conclusion requires accuracy relative to the counterfactual W_d.

C.7 Discounting and the values required by the analysis

With discounted returns, $Q^{\bar{\pi}}$ already includes the environment discount factor. Internal refinement is instantaneous in this model, so no additional γ^d factor belongs in Equation (24). Nevertheless, changing γ can change the ordering and increments of W_d; it does not generally cancel from their differences. If computation consumes environment time or causes delayed actions, that delay requires a different value/cost model.

The first-crossing assumption concerns increments of the counterfactual action values W_d. The shared-target predictions are characterized separately in Section A.9.

D Tasks

The benchmark combines tasks with different observations, action spaces and opportunities for useful refinement. Table 8 summarizes the twelve-task set; the language task is specified below.

D.1 Language Task: GPT-2 on GLUE-RTE

GPT-2 on GLUE-RTE tests adaptive refinement outside sequential control. The observation is an encoded text pair and the action is a binary label, allowing us to study allocation with a frozen pretrained encoder and a small action space.

Task. GLUE-RTE [Wang et al., 2018a] is binary recognizing-textual-entailment: given a (premise, hypothesis) token sequence, predict `entailment` or `not_entailment`. We use the standard 2,490-example train split and the 277-example dev split as evaluation; the test split has hidden labels and is not used. Each example is a single-step episode: the agent observes the tokenized pair, picks a label, receives reward $+1$ if correct and 0 otherwise and the episode ends. This collapses RL to bandit-style classification but preserves AIC's compute-allocation question: how many refinement steps to spend on this example before committing to a label.

Table 8: **Tasks.** Every benchmark setting, including DistShift, uses five seeds; Section S summarizes the protocols of the additional studies.

Task	Domain	Source	Role
HalfCheetah-v4	Continuous control	Gym MuJoCo	anchor
ManiSkill PickCube-v1	Robotic manipulation	ManiSkill3	anchor
ManiSkill StackCube-v1	Sequential manipulation	ManiSkill3	anchor
AntMaze (main)	Maze navigation	D4RL/Gymnasium-Robotics	anchor
AntMaze (low-het.)	Maze nav. (low-heterogeneity)	D4RL variant	low-het.
GPT-2 on GLUE-RTE	Entailment classification	GLUE-RTE on GPT-2	anchor
MiniGrid-DistShift2	Gridworld dist. shift	MiniGrid	sat. ctrl.
MiniGrid-FourRooms	Gridworld FourRooms	MiniGrid	anchor
OGBench cube-double-play-v0	Goal-cond. multi-cube	OGBench	goal-cond.
OGBench scene-play-v0	Goal-cond. scene	OGBench	goal-cond.
OGBench puzzle-4x4-play-v0	Goal-cond. puzzle	OGBench	goal-cond.
RareEvent-StackCube-Misalign	Rare-event manipulation	ManiSkill3 (slip/misalign)	rare

Encoder. The encoder is GPT-2-small with 124M frozen parameters [Radford et al., 2019]; the language-model head is unused and gradients do not update GPT-2. The GPT-2 BPE tokenizer represents each input as `<premise> <sep> <hypothesis>`, padded or truncated to 256 tokens. We take the 768-dimensional hidden state at the final non-padding token of the last layer and project it through a trained linear layer to the 256-dimensional refinement representation.

Refinement and heads. The shared MLP applies $z_d = z_{d-1} + \mathrm{MLP}(z_{d-1})$ to the projected representation. Each depth has a two-class softmax action head π_d, a scalar value head V_d and a sigmoid halting head h_d. Every value head receives the same GAE target, which reduces here to the one-step Monte Carlo return. The selected label and stopping depth enter the joint PPO surrogate.

Evaluation and aggregation. Each checkpoint is evaluated ten times on the 277-example dev split, with actions sampled from the selected depth's action distribution and its accuracy is the mean over these passes. Tables report the mean $\pm$ standard deviation of the per-seed accuracies over five seeds. One dev example corresponds to $1/277 \approx .0036$ accuracy, so the $B = 8$ gain of $+.0023$ in Table 1 amounts to 0.64 examples per seed. The confidence-threshold baseline selects its entropy threshold on a held-out 249-example slice of the training split, which is excluded from policy training.

Allocated computation. AIC allocates refinement after text encoding: GPT-2's forward pass is unchanged and the task makes one classification decision rather than generating tokens autoregressively. The benchmark checkpoints use a mean depth of 3.07; the timing study in Section Q uses a separately trained checkpoint at $\lambda_c = .003$ whose mean dev-set depth is 4.08.

Critical-state annotation. The decision unit is the whole premise–hypothesis pair rather than individual tokens, so critical-state annotation labels whole examples.

D.2 POGP Comparison Environments

Table 2 uses the Gym MuJoCo locomotion tasks HalfCheetah-v4, Walker2d-v4, Ant-v4 and Hopper-v4 with POGP's training and evaluation protocol and unnormalized returns, so these runs are separate from the benchmark's HalfCheetah task. POGP's protocol evaluates the first 100 environment steps of each episode, so its returns are about a tenth of the usual 1000-step returns and include the acceleration from rest at the start of every episode. AIC is attached to POGP's diffusion policy as in Section R: its halting head reads the denoiser's hidden features, one iteration is one denoising step in both methods and both methods share the eight-step cap.

E Relation to ACT, PonderNet and Metacontrol

Ghugare et al. [2026] study fixed-depth iterative policies and identify adaptive depth within one policy as an open direction. AIC learns that decision using established stopping and metacontrol ideas.

Established stopping mechanisms. ACT makes recurrent computation input-dependent through a task objective and a ponder cost [Graves, 2016]. PonderNet learns a distribution over iterative prediction depths [Banino et al., 2021], while metacontrol uses RL to choose the identity and number

of internal expert queries [Hamrick et al., 2017]. Adaptive depth, probabilistic stopping and reward-sensitive computation therefore precede AIC.

Policy formulation and optimization. ACT combines intermediate outputs using accumulated halting mass and a ponder penalty. PonderNet uses conditional stopping probabilities, an expected prediction loss and a KL penalty toward a geometric stopping prior. AIC uses the same general conditional-stopping construction to select one depth-specific action policy. Its joint action/stopping probability enters the PPO ratio, with an expected-depth penalty and auxiliary value supervision (Section 3.1). Earlier metacontrol also learns computation decisions through RL; AIC applies stopping directly inside the action policy's refinement process. A constant cost permits state-dependent stopping in both settings (Section C.2).

Allocation analysis and empirical attribution. Theorem 1 compares an ideal state-dependent allocator with state-independent stopping at equal expected depth, under a fixed state distribution and explicit value-profile assumptions. The experiments separate the effects of stopping, supervision, data collection and refinement from the narrower effect of reassigning equal computation across states and Section F analyzes the task-specific factorial effects.

F Factorial Details

The four configurations in Table 5 vary adaptive stopping and auxiliary per-depth value supervision. The auxiliary per-depth loss is distinct from the ordinary value baseline used to estimate PPO advantages and every configuration is an independently trained configuration.

Conditional effects and interaction. For R_{as} as defined in Section 4.2.3, the interaction is

$$I = (R_{11} - R_{01}) - (R_{10} - R_{00}) = (R_{11} - R_{10}) - (R_{01} - R_{00}). \tag{32}$$

From the configuration means, $I = .003$ for PickCube, .006 for AntMaze, .004 for Memory, .002 for DoorKey, $-.005$ for StackCube and zero for RareEvent. These interactions are small relative to the main effects but not uniformly zero, so we give both conditional effects rather than assume additive effects.

From configuration changes to allocation. An adaptive setting can change mean depth, the distribution of depths, the action heads learned during training and which states receive computation. The stopping effect therefore combines these changes. In our factorial the non-adaptive configurations execute one refinement per decision and the adaptive configurations execute 1.8–3.9 on average (Table 5), so each stopping effect includes additional computation as well as state-dependent assignment. An allocation-specific comparison would replace the non-adaptive schedule with a separately tuned state-independent schedule at the adaptive configuration's mean computation, keeping the encoder, refinement cost and training update budget fixed. Changing the supervision factor preserves the ordinary advantage baseline and the parameter count, since the value heads remain in the network and only their loss weight is set to zero. Treating configurations as independent, the PickCube stopping effects have standard errors of .0032 and .0029 and its supervision effects .0035 and .0025; the RareEvent stopping effects of .002 lie within one standard error, whereas its supervision effects of .031 exceed six.

G Halting Baselines

Untrained stopping rules. The confidence rule stops when policy entropy falls below a validation-selected threshold. Action agreement compares adjacent action distributions, while the value-gain rule compares adjacent value predictions; both need a previous depth, so their earliest stop is depth two. A mean depth of exactly 2.00 in Table 10 therefore means that the rule stopped at depth two on every decision. This holds for both rules on DistShift and RareEvent, where the two rules coincide; their RareEvent reward of .4198 measures the depth-two action head on every state. The random schedule stops at each depth $d < d_{\max}$ with a state-independent probability β, so $q(d) = \beta(1-\beta)^{d-1}$ for $d < d_{\max}$, the cap receives $(1-\beta)^{d_{\max}-1}$ and $\mathbb{E}[\tau] = (1-(1-\beta)^{d_{\max}})/\beta$. For each task, β maximizes validation return under a golden-section search over $\log\beta \in [\log 10^{-3}, 0]$; Table 10 lists the selected values. The schedule is therefore a tuned state-independent stopping law, the empirical counterpart of the baseline class in Theorem 1, without matching AIC's mean depth. With $\beta = 1$ it executes one refinement on every decision,

as on DistShift, low-heterogeneity AntMaze and RareEvent. All untrained rules are applied to the trained AIC network at $B = 8$ and replace only its halting head.

Jointly trained baselines. Table 9 specifies the two jointly trained baselines alongside AIC. ACT-ponder follows Graves [2016]: its halting units accumulate halting mass until it exceeds $1 - \varepsilon_{\mathrm{ACT}}$ with $\varepsilon_{\mathrm{ACT}} = .01$, the action distribution is the halting-weighted mixture of the depth-specific heads and the ponder cost $N + R$ enters the PPO loss with weight .01. The learned non-VoC baseline keeps AIC's architecture and evaluation rule and changes only the halting objective: its stopping probabilities are excluded from the PPO ratio and a two-layer halting MLP reads a stop-gradient copy of z_d and is trained by binary cross-entropy toward $\mathbf{1}\{\mathrm{KL}(\pi_d \,\|\, \pi_{d+1}) < .01\}$ on the same rollout states. The name records that this stopping signal is learned without value-of-computation credit: no return or advantage reaches its halting decision. All three methods can vary depth across states, including with a constant per-step cost and we evaluate their recorded reward and mean depth together.

Table 9: **Jointly trained halting methods.** All three share the encoder, refinement block, depth-specific action heads and value heads and use five seeds at $B = 8$.

	AIC	ACT-ponder	Learned non-VoC
Halting input	z_d	z_d	stop-gradient copy of z_d
Halting head	linear, one per depth $d < d_{\max}$	linear, shared across depths	two-layer MLP ($256 \to 160 \to 1$), shared
Halting objective	joint PPO ratio over (τ, a)	task loss through the halting-weighted action mixture	cross-entropy toward $\mathbf{1}\{\mathrm{KL}(\pi_d \| \pi_{d+1}) < .01\}$
Stopping in PPO ratio	yes	through the mixture likelihood	no
Computation penalty	$\lambda_c \mathbb{E}[\tau]/d_{\max}$, $\lambda_c = .01$	ponder cost $N + R$, weight .01	none
Per-depth value supervision	yes, $\lambda_V = .5$	yes, $\lambda_V = .5$	yes, $\lambda_V = .5$
Evaluation rule	first $h_d > .5$, else cap	halt once accumulated mass exceeds $1 - \varepsilon_{\mathrm{ACT}}$	first $h_d > .5$, else cap
Parameters	2.071M	2.069M	2.110M

Task-dependent tradeoffs. RareEvent has small truncation effects and larger supervision effects. On FourRooms, ACT has both greater mean depth and higher reward than AIC, placing it at a different return–compute operating point.

H Refinement Depth and Computation Accounting

Mean depth measures how many refinement steps a policy executes per decision. Table 10 compares that quantity across the twelve benchmark tasks. End-to-end timing additionally includes the encoder and control-flow overhead (Section Q).

Table 10: **Mean refinement depth at B=8 on all twelve tasks.** Mean steps per state for the seven stopping methods; trained-method entries carry the seed standard deviation as a subscript and β is the random schedule's per-task stopping probability (Section G). Fixed-K controls execute exactly K refinements. ACT and non-VoC halt deeper than AIC on eleven tasks; HalfCheetah is the exception. The final row is the unweighted task mean.

	Trained			Untrained / heuristic				
Task	AIC	ACT	non-VoC	Conf.	Agr.	Val.-g.	Rand.	β
HalfCheetah-v4	$3.04_{\pm.07}$	$2.94_{\pm.08}$	$2.93_{\pm.08}$	2.90	2.85	3.24	2.00	.498
ManiSkill PickCube-v1	$3.47_{\pm.09}$	$3.80_{\pm.14}$	$3.85_{\pm.12}$	3.51	3.79	3.38	3.11	.304
ManiSkill StackCube-v1	$3.77_{\pm.11}$	$4.80_{\pm.17}$	$4.75_{\pm.15}$	3.86	3.42	3.86	3.55	.255
AntMaze (main)	$3.15_{\pm.08}$	$3.54_{\pm.12}$	$3.65_{\pm.13}$	3.07	2.85	3.14	2.12	.469
AntMaze (low-het.)	$2.24_{\pm.06}$	$2.64_{\pm.09}$	$2.56_{\pm.08}$	2.16	2.22	2.22	1.00	1
GPT-2 on GLUE-RTE	$3.07_{\pm.06}$	$3.42_{\pm.09}$	$3.33_{\pm.08}$	3.23	3.04	2.93	1.30	.769
MiniGrid-DistShift2	$1.00_{\pm.00}$	$1.04_{\pm.03}$	$1.01_{\pm.01}$	1.00	2.00	2.00	1.00	1
MiniGrid-FourRooms	$2.43_{\pm.12}$	$6.82_{\pm.21}$	$6.19_{\pm.19}$	2.28	2.46	2.26	1.35	.741
OG cube-double-play	$5.19_{\pm.14}$	$6.68_{\pm.19}$	$6.67_{\pm.17}$	5.27	2.84	4.93	7.59	.0151
OG scene-play	$5.05_{\pm.13}$	$6.23_{\pm.18}$	$6.22_{\pm.16}$	4.98	2.78	5.13	7.19	.0307
OG puzzle-4x4-play	$5.60_{\pm.15}$	$7.20_{\pm.16}$	$7.26_{\pm.15}$	5.64	2.49	5.26	7.92	.0029
RareEvent-Misalign	$1.86_{\pm.05}$	$1.99_{\pm.04}$	$1.99_{\pm.04}$	1.50	2.00	2.00	1.00	1
Mean (12 tasks)	3.32	4.26	4.20	3.28	2.73	3.36	3.26	—

Table 10 gives average refinement use; Sections L and M examine how depth is distributed across states.

Combining depth with reward. Figure 3 combines the mean depths with the seven-task rewards in Table 6. A fixed baseline's depth is the seed average of its executed depths $\min(K_{\text{train}}, 8)$. For AntMaze and RareEvent, the fixed rewards pool training depths 2 and 8 (one and four seeds) and 1 and 8 (two and three seeds), giving mean executed depths of 6.8 and 5.2. On FourRooms, AIC's lower mean depth accompanies lower mean reward than the trained alternatives.

I Hyperparameters and Sensitivity on Failure Tasks

We vary the computation and value-loss weights to test whether the two default-setting failures persist.

Default configuration. The benchmark draws on MuJoCo/Gym [Brockman et al., 2016, Todorov et al., 2012], ManiSkill3 [Tao et al., 2025], D4RL-style navigation [Fu et al., 2020], MiniGrid [Chevalier-Boisvert et al., 2018], OGBench [Park et al., 2025] and GLUE-RTE [Wang et al., 2018a]. The twelve-task benchmark uses Adam, learning rate 3×10^{-4}, $\gamma = .99$, GAE $\lambda_{\text{GAE}} = .95$, PPO clip $\eta = .2$, ten PPO epochs, batch size 64, four parallel environments, hidden dimension 256, $d_{\max} = 8$, $\lambda_c = .01$ and $\lambda_V = .5$. Anchor tasks use seeds 41–45 and DistShift also uses five seeds. The two loss weights were selected on PickCube and shared across all twelve benchmark tasks. λ_c is expressed relative to the extrinsic-reward scale; a single numerical value therefore need not represent the same tradeoff across tasks. Reward normalization and a dual variable for an average-compute constraint are possible extensions.

Loss-weight selection on PickCube. Sweeping the compute weight over $\{0, .001, .005, .01, .05, .1\}$ gives PickCube rewards of .4897–.4904 and sweeping the value weight over $\{0, .1, .5, 1, 2\}$ gives .4898–.4925, so PickCube is insensitive to both weights. We then test whether these weights transfer to FourRooms and Puzzle.

Failure-task sweeps. We sweep $\lambda_c \in \{.001, .003, .01, .03, .1\}$ and $\lambda_V \in \{.25, .5, 1, 2\}$ on FourRooms and Puzzle, comparing each setting with a fixed-depth baseline matched to AIC's mean inference depth. FourRooms stays negative across the grid; tuning narrows its default deficit from

Table 11: FourRooms: difference from the mean-depth-matched fixed baseline in percentage points of normalized reward (one point is .01 in the units of Table 1), mean ± standard error over five seeds.

λ_c	$\lambda_V = .25$	$\lambda_V = .5$	$\lambda_V = 1$	$\lambda_V = 2$
.001	$-.9 \pm .3$	$-.7 \pm .3$	$-.8 \pm .4$	$-1.0 \pm .4$
.003	$-1.0 \pm .3$	$-.9 \pm .3$	$-1.0 \pm .3$	$-1.2 \pm .4$
.010	$-1.7 \pm .4$	$-1.8 \pm .4$	$-1.6 \pm .4$	$-1.8 \pm .5$
.030	$-1.8 \pm .4$	$-1.9 \pm .4$	$-1.8 \pm .5$	$-2.0 \pm .5$
.100	$-2.0 \pm .5$	$-2.1 \pm .5$	$-2.0 \pm .5$	$-2.1 \pm .5$

$-1.8 \pm .4$ to $-.7 \pm .3$. On Puzzle (Table 12), the differences are positive, $+.6$ to $+.9$ percentage points, for every λ_c when $\lambda_V \in \{1, 2\}$ and negative for $\lambda_V \leq .5$, including $-.5 \pm .3$ at the default setting.

Table 12: Puzzle: difference from the mean-depth-matched fixed baseline in percentage points of normalized reward, mean ± standard error over five seeds.

λ_c	$\lambda_V = .25$	$\lambda_V = .5$	$\lambda_V = 1$	$\lambda_V = 2$
.001	$-.6 \pm .3$	$-.3 \pm .3$	$+.8 \pm .3$	$+.9 \pm .3$
.003	$-.5 \pm .3$	$-.3 \pm .3$	$+.9 \pm .3$	$+.8 \pm .3$
.010	$-.7 \pm .3$	$-.5 \pm .3$	$+.7 \pm .3$	$+.8 \pm .3$
.030	$-.8 \pm .4$	$-.6 \pm .3$	$+.7 \pm .3$	$+.7 \pm .4$
.100	$-1.0 \pm .4$	$-.8 \pm .4$	$+.6 \pm .4$	$+.6 \pm .4$

Puzzle is thus sensitive to the value weight, while FourRooms is a persistent limitation within the tested range.

J Privileged Reference and Remaining Headroom

The privileged reference estimates how much performance remains available when depth selection can use task-specific information. It selects $\min(\hat{\kappa}(s), B)$ using torque percentiles for HalfCheetah; contact and phase indicators for manipulation; intersection proximity for AntMaze; a label-margin probe for RTE; a slip indicator for RareEvent; and goal-distance bins for OGBench. This is a heuristic reference under a common depth cap. Its gap recovery measures progress toward that reference, rather than an optimal-policy bound or a matched-work comparison. The reference runs AIC's trained network at each budget and changes only the stopping rule, so at $B = 1$ it coincides with AIC. Gap recovery is measured relative to this reference and is undefined when its reward does not exceed the fixed baseline's reward; Table 14 lists the budget-specific reference rewards.

Table 13: Headroom diagnostic on PickCube, AntMaze, Memory and DoorKey: fraction of the privileged-reference gap recovered.

Setting	Gap recovered
Default halting entropy	6.1%
$2\times$ halting entropy	6.8%
$4\times$ halting entropy	7.4%
Privileged future-benefit probe	29.8%

The future-benefit probe uses realized future benefit, which is unavailable at deployment; its 29.8%, against at most 7.4% from higher halting entropy, points to predicting useful refinement from deployment-time information as the main opportunity. On an eight-task subset evaluated separately, AIC recovers 5.4% of the gap on average.

Localized refinement on AntMaze. With five fresh seeds, AIC recovers 6.3% of the AntMaze reference gap, while the best local-mask variant, which restricts refinement spatially, recovers 14.2% at the same mean depth and improves normalized reward by 2.6 percentage points; these runs are independent of the 8.1% recovery in Table 1. Spatially restricted refinement therefore appears to reduce interference from irrelevant features.

K Full Nominal-Budget Comparison (12 tasks)

Table 14 gives every nominal-budget comparison in the twelve-task benchmark. The cap bounds the available depth, so mean computation can still differ between methods; Table 15 gives AIC's mean depth at each budget. For a fixed checkpoint trained at depth K_{train}, evaluation executes $K_{\text{exec}} = \min(K_{\text{train}}, B)$ refinements and uses the action head at that executed depth. The parenthesized depths identify checkpoint training depths selected across seeds, with superscripts giving the number of seeds that selected each depth when more than one was selected. Thus $(1^{(4)}, 4^{(1)})$ at $B = 1$ means that four seeds selected a depth-one checkpoint and one seed a depth-four checkpoint, both executing one refinement; $(4^{(3)}, 8^{(2)})$ at $B = 4$ similarly executes four. Each budget's baselines come from that budget's training sweep (Section 4), so a depth-one checkpoint selected at $B = 8$ is a different training run from the depth-one checkpoints at $B = 1$. Derived differences, relative gains and reference-gap recoveries in Tables 1 and 14 are computed from the displayed means, before rounding the derived quantity; each budget uses its own reference reward, listed in the Ref. column. Against the strongest fixed baseline at any budget up to B, AIC has 6, 8 and 7 positive differences at $B = 2$, 4 and 8, compared with 8, 10 and 9 against the per-budget baselines; low-heterogeneity AntMaze and RareEvent account for the difference at every budget because their depth-one baselines lead throughout.

Puzzle across budgets. At the default loss weights, AIC improves on Puzzle at $B \in \{1, 2, 4\}$ and trails at $B = 8$; a larger value weight reverses the $B = 8$ loss (Section I), so the budget pattern reflects the training configuration as well as adaptive depth.

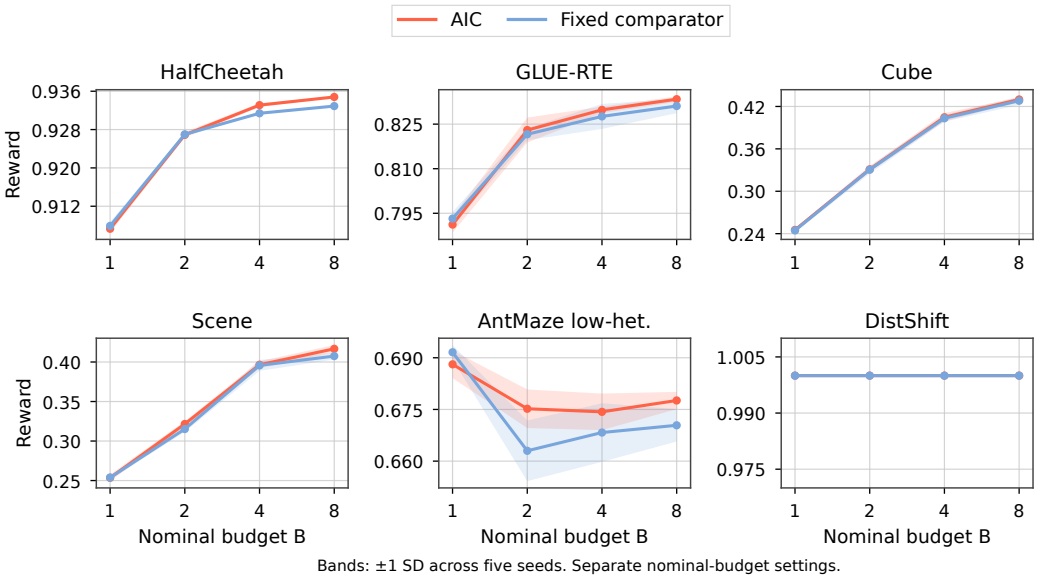


Figure 4: **Performance across nominal budgets on the remaining six benchmark tasks.** Points are five-seed means and shaded bands are ± 1 standard deviation from Table 14. The budget convention and fixed-baseline definitions match Figure 2. DistShift is saturated at every budget; the other tasks respond differently to a larger budget.

L Critical-State Methodology

Selecting states for early exit. For each AIC seed trained at $B = d_{\max} = 8$, we run N evaluation episodes and record the selected depth τ_t at each visited state. With $u_{.75}$ the seed's 75th-percentile depth, we re-evaluate the policy with every decision satisfying $\tau_t \geq u_{.75}$ forced to depth one. Ties at the cutoff can select more than a quarter of decisions, so Table 16 lists the realized selected fractions.

Random-state control. For each seed, we force a uniformly sampled set of decisions of the same size to depth one and average the drop over ten samples. This control matches the number of interventions but not the amount of removed computation.

Estimand. The drop is $R(\pi) - R(\pi_A)$, where $R(\pi)$ is the return of the unmodified policy and $R(\pi_A)$ the return after applying an early-exit rule A. Because the modified policies can visit different states after their first altered action, targeted-minus-random includes downstream effects on action quality and occupancy.

Equal-work control. On each seed's reference set, we record the unmodified depths τ_i and the removed work $M(A) = \sum_{i \in A}(\tau_i - 1)$ of the targeted set. The equal-work baseline draws decisions uniformly at random, without replacement and independently of task annotations or estimated difficulty, until the cumulative removed work reaches $M(A)$; the last drawn decision is truncated to the intermediate depth that makes the totals equal. We average its drop over ten draws per seed and on complete rollouts the achieved mean work of the two intervened policies differs by at most 3%. This control matches removed computation rather than the number of interventions; Table 16 gives the realized selected fractions, the removed refinements per intervened decision and the drops.

Depth reassignment. Depth reassignment would complement the equal-work control: permuting the recorded depth multiset over a common set of saved states preserves total refinement while removing the association between state and depth. Its effect can be measured on the saved states under a common continuation policy, which isolates one decision at fixed occupancy, or with full rollouts, which include changes in future state visitation; in both cases the comparison is paired by seed and evaluation states.

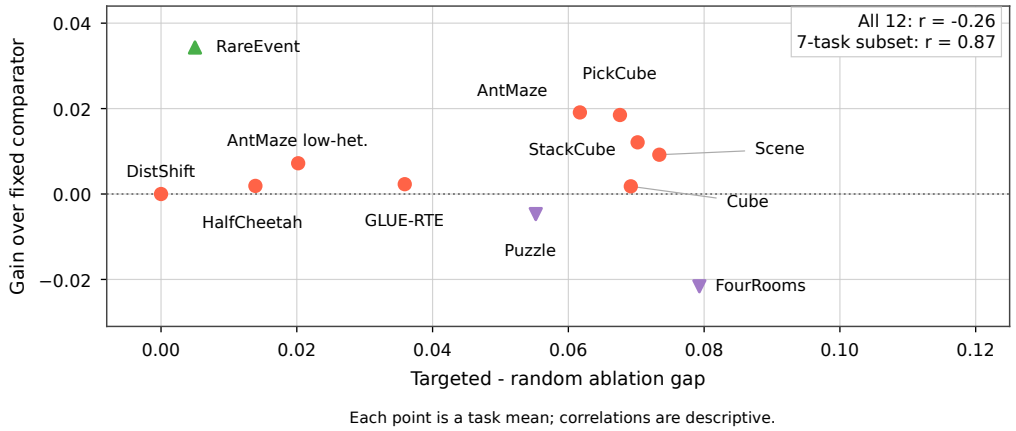


Figure 5: **Allocation diagnostic and fixed-baseline gain.** Each point is a task mean from the twelve-task benchmark. The seven-task subset (DistShift, HalfCheetah, low-heterogeneity AntMaze, RTE, AntMaze, PickCube, StackCube) has Pearson $r = 0.87$; including Cube and Scene gives $r = 0.62$ and all twelve points give $r = -0.26$. The horizontal axis is count-matched truncation sensitivity; the correlation depends on the included task set.

M Critical-State Annotations and a PickCube Case Study

Semantic annotations test whether greater depth coincides with recognizable task events. The proxies are torque-based recovery in HalfCheetah, flagged when joint torque exceeds the 90th percentile of the evaluation episodes; contact, gripper closure, lifting and stacking transitions in ManiSkill; junction proximity in AntMaze, within 1.5 m of a corridor junction; slip recovery in RareEvent, flagged when cube–gripper relative speed exceeds 2 cm/s; and goal-distance thresholds in OGBench, with bins at 0.05, 0.1 and 0.2 m. In ManiSkill, contact begins when either finger reports a contact force above 0.5 N, grasp requires both fingers in contact with gripper closure above 80%, lift begins when the cube rises 2 cm above the table and stabilization/recovery covers the remaining lifted steps, including regrasps after a slip. The PickCube phase study uses the five benchmark AIC checkpoints at $B = 8$ with 50 evaluation episodes per seed, so its population matches the PickCube depth in Table 10.

AIC spends the most refinement and gains the most from it, around contact and recovery, while the approach phase stays shallow.

N Task-Level Analysis and Cross-Task Association

Across the six factorial tasks, stopping effects exceed supervision effects on PickCube, AntMaze, Memory, DoorKey and StackCube, where the adaptive configurations also execute more refinements, while RareEvent shows the reverse (Section F).

Among the remaining tasks, FourRooms stays below its mean-depth-matched baseline throughout the sweep, Puzzle improves with stronger value supervision, DistShift is a saturated tie and HalfCheetah, low-heterogeneity AntMaze and RTE show small positive differences. RareEvent pairs a shallow mean depth of 1.86 with a large supervision effect.

Association across tasks. Figure 5 relates each task's targeted-minus-random drop to its gain over the per-budget fixed baseline. The correlation is $r = .87$ on a seven-task subset, .62 after adding Cube and Scene and $-.26$ over all twelve tasks, so even its sign depends on the task set. The count-matched difference also mixes state choice with removed work and the gain compares separately trained policies. With the equal-work difference of Table 16 instead, the correlation is $r = .39$ over all twelve tasks and $r = .75$ on the seven-task subset. Our conclusions rest on the per-task evidence.

O Limitations and Failure Modes

Predictability and non-monotone refinement. State-dependent computation helps when useful additional depth can be predicted from the current representation. Low immediate value gain can precede a larger later gain, invalidating a first-crossing heuristic without diminishing returns. The continuation recursion in Section C handles this possibility; the shared-target critics do not verify its counterfactual value assumptions.

Shared targets and intervention interpretation. Shared GAE targets do not identify forced-depth counterfactual returns. The stated forward graph also makes a frozen policy invariant to value-output replacement (Proposition 4); such a replacement cannot test its stopping mechanism without changing additional parts of the policy. Count-matched targeted/random truncation can remove different amounts of compute; the equal-work control in Table 16 matches removed refinements and reduces the mean targeted-minus-random difference from .0463 to .0118. A depth-reassignment control would further separate the association between state and depth.

Reward scale and operating point. The computation penalty depends on reward units. Four-Rooms remains below its matched baseline throughout the tested grid, whereas Puzzle improves under stronger value supervision. The default setting defines one operating point, with task-specific sensitivity to its loss weights.

Deployment versus training. Training takes $1.22\times$ the Fixed-8 time. End-to-end speedup is smaller when an expensive encoder dominates, as shown by GPT-2/RTE's 7.8% latency reduction despite 49% fewer refinement steps. A depth cap bounds the number of refinement steps, but a hardware-specific latency guarantee also requires bounds on the encoder, heads, scheduling and execution time.

Broader impact. Adaptive computation can lower inference latency and energy for control policies when refinement is a substantial share of the per-decision cost; for policies dominated by a large frozen encoder the saving is small (Section Q) and training costs $1.22\times$ the Fixed-8 time. Like other RL methods, AIC can be applied to systems in which errors cause physical harm and state-dependent stopping adds a specific failure mode: halting too early in rare states that the training data cover poorly. Deployments in safety-critical settings should monitor stopping depth and can enforce a minimum depth where errors are costly.

Scale and reproducibility. The experiments use small refinement modules and simulated environments; scaling to large backbones and real robots remains untested. The privileged reference is a diagnostic rather than a certified upper bound. The factorial, timing and sensitivity studies use run sets separate from the twelve-task benchmark; the code release includes their seed-level records and compute accounting.

P Memory Task Extensions

Memory and DoorKey extend AIC to decisions that depend on past observations, which require a history representation in addition to within-decision refinement (Table 18).

Tasks. We use two MiniGrid environments with partial observability [Chevalier-Boisvert et al., 2018]. In *Memory*, the agent enters a corridor where a colored cue is shown, then proceeds to a junction where it must select the path matching the cue; the cue is no longer visible at the junction, so the policy must internalize it across timesteps. In *DoorKey*, the agent must locate a key, pick it up, traverse to a locked door and use the key; the key and door are typically not co-visible in the agent's egocentric view. Both tasks use sparse completion rewards and require the representation to retain relevant history.

History representation. Both tasks use a single-layer GRU with 256 hidden units over the CNN embedding of each observation and the previous action. The GRU state ψ_t is updated once per environment step and $z_0 = f_{\text{enc}}(\psi_t)$; refinement does not modify ψ_t and z_τ is not fed back into the GRU. Training uses truncated backpropagation through time over 32 environment steps and ACT-ponder, learned non-VoC and Fixed-8 share the same GRU and encoder.

Critical-state precision and recall. A decision is high-depth when τ is at or above its seed's 75th-percentile depth. The critical states are the simulator-annotated junction step in Memory, where

the cue must be recalled and the key-pickup and door-opening steps in DoorKey. Precision is the fraction of high-depth decisions within one step of a critical state and recall is the fraction of critical states whose decision is high-depth. For the benchmark rows of Table 18, a decision is critical when it lies within one step of contact onset, grasp closure, lift-off or slip recovery in PickCube, which covers 23% of decisions, or within 1.5 m of a corridor junction in AntMaze, which covers 15%; precision and recall are computed over these critical decisions. These event windows are narrower than the phases of Table 17: treating whole grasp, lift and stabilization phases as critical would give PickCube a precision of .81 and a recall of .65 instead of .49 and .74.

Results. The factorial has larger stopping-configuration effects on both tasks (Section N), where the adaptive configurations execute 3.3–3.5 refinements against one for the non-adaptive configurations. On Memory, AIC obtains reward 0.5715 at mean depth $\bar{d} = 3.31$, a $+0.021$ increase over Fixed-8 with 0.31 fewer steps than ACT-ponder. On DoorKey, it obtains 0.5470 at $\bar{d} = 3.36$, again with lower mean depth. The targeted-minus-random gaps are $+0.056$ and $+0.059$, respectively. Against simulator annotations, precision and recall are $(0.42, 0.79)$ at memory-query positions and $(0.44, 0.85)$ at key-pickup positions: most critical states receive high depth, while fewer than half of the high-depth decisions fall at critical states.

Memory and DoorKey are not part of the twelve-task benchmark or the cross-task correlation.

Q Training Cost and Measured Inference Latency

The timing study separates the cost of training all intermediate heads from the savings of stopping early at inference. It also distinguishes refinement count from end-to-end runtime. All timings use a single NVIDIA A100 80 GB GPU with an AMD EPYC 7763 host CPU, PyTorch 2.3 and CUDA 12.1. Inference latency is the median over 1,000 decisions after 100 warm-up decisions at batch size one; the control and robotics rows use the five benchmark checkpoints at $B = 8$ and the RTE row uses the timing checkpoint described in Section D.1. The 31–44% reductions are therefore per-decision latency, not throughput.

The training comparison holds hardware, batch configuration, update count and training setup fixed; both PickCube runs use ten million environment steps. Every refinement depth is evaluated during optimization for value supervision, including depths beyond the sampled action. Fixed-8 evaluates the same eight refinement steps and value heads, so the additional AIC time comes from the halting heads, the survival and joint log-probability computations and the per-sample gather of the selected depth's action head, which our implementation does not fuse into one kernel. At inference, total work includes the encoder, executed refinements, heads and control-flow overhead. Refinement count and total FLOPs therefore decrease by the same proportion only when the remaining costs are negligible. For an additive per-action cost model $T(d) = T_{\mathrm{enc}} + dT_{\mathrm{ref}} + T_{\mathrm{other}}$ with constant marginal refinement time, the fractional time saved by reducing mean depth from d_0 to d_1 is

$$\frac{T(d_0) - T(d_1)}{T(d_0)} = \frac{(d_0 - d_1)T_{\mathrm{ref}}}{T_{\mathrm{enc}} + d_0 T_{\mathrm{ref}} + T_{\mathrm{other}}}. \tag{33}$$

Fitting this model to Table 19 gives $T_{\mathrm{ref}} \approx 0.19$ ms and $T_{\mathrm{enc}} + T_{\mathrm{other}} \approx 7.92$ ms for RTE, so refinement accounts for about 16% of Fixed-8's per-decision time and $T_{\mathrm{ref}} \approx 0.07$–$0.08$ ms with 0.25–0.39 ms of fixed overhead for the control and robotics policies. The encoder and other fixed costs explain how a large refinement reduction can produce a smaller end-to-end change. The experiments in this paper used about 9,800 A100 GPU-hours in total, plus about 1,200 GPU-hours of preliminary and discarded runs.

R Base-Policy Transfer Details

Table 7 summarizes the eight-task transfer to DiffusionQL, DPPO, D3P, DSAC-D and IDQL [Wang et al., 2022, Ren et al., 2025, Yu et al., 2025, Liu et al., 2025, Hansen-Estruch et al., 2023]. Each policy family defines its own refinement operation, intermediate-action decoder and training update, so one step can have a different computational cost across families. Table 20 specifies how AIC is attached to each family and Table 21 lists the full results. In every family one refinement is one denoising step of an eight-step sampler, the halting head reads the denoiser's hidden features at that step concatenated with the state embedding and the intermediate action is the one-step prediction

$\hat{a}_0$ of the clean action from the partially denoised sample. DPPO and D3P are trained on-policy, so their halting heads use the joint PPO ratio of Section 3 with the denoising chain as internal refinement. DiffusionQL and IDQL are trained offline and DSAC-D off-policy, where no rollout ratio is available; their halting heads instead maximize $\mathbb{E}_{\tau\sim q_\theta}[Q(s,\hat{a}_{0,\tau})] - \lambda_c \mathbb{E}[\tau]/d_{\max}$ with a score-function gradient through q_θ, holding the family's own critic fixed during the halting update. D3P's own adaptive step allocation is disabled in both its Fixed and AIC variants, so the comparison isolates the AIC halting head.

Datasets and discrete actions. DiffusionQL and IDQL train on fixed datasets of one million transitions per task: the D4RL `halfcheetah-medium-expert-v2` dataset, uniformly subsampled from its two million transitions, the full D4RL `antmaze-medium-diverse-v2` dataset, the OGBench `scene-play-v0` and `puzzle-4x4-play-v0` datasets and, for ManiSkill, MiniGrid and RareEvent, which have no standard offline datasets, transitions collected by the benchmark Fixed-8 policy with Gaussian action noise of standard deviation 0.2, or 20% uniformly random actions on FourRooms. The D4RL `-v2` data were recorded in the mujoco-py version of HalfCheetah; HalfCheetah-v4 keeps its observation and action spaces, so we evaluate the trained policies in HalfCheetah-v4 and normalize returns with D4RL's reference scores. DPPO and D3P fine-tune online from a behavior-cloned initialization on the same data and DSAC-D trains online from its replay buffer. For FourRooms, whose seven actions are discrete, every family denoises a seven-dimensional continuous vector and executes its argmax.

S Experimental Protocol Summary

The main benchmark covers twelve tasks at four nominal budgets with five seeds each and the POGP comparison covers four environments with five seeds under an eight-step cap. The transfer study covers eight tasks in each of five policy families; the memory-task comparisons and the stopping-by-supervision factorial, four configurations on six tasks, use their own runs. The sensitivity study covers five-by-four loss-weight grids for FourRooms and Puzzle. The truncation interventions reuse the frozen benchmark checkpoints and average ten random subsets per seed for both the count-matched and equal-work controls. The local-mask comparison uses five fresh seeds and the entropy/probe and timing studies use their own runs.

Uncertainty. Each caption states its uncertainty convention. The benchmark, the halting-baseline, transfer and POGP comparisons and the data-source and stopping and supervision ablations give standard deviations over five seeds and the sensitivity grids give standard errors over five seeds. RTE accuracies average ten sampled evaluation passes per seed before the across-seed statistics (Section D.1). Correlations are computed from task means.

Protocol details. The fixed-baseline selection rule and seed counts (Sections 4 and K), the random-stopping law (Section G), the halting-baseline specifications (Table 9), factorial schedules and depths (Table 5), annotation thresholds and populations (Section M), memory and transfer interfaces (Sections P and R), budget-specific reference rewards (Table 14) and timing hardware and durations (Section Q) accompany the corresponding results. Section O discusses the limitations.

Table 14: Full comparison at nominal $B \in \{1, 2, 4, 8\}$. Means $\pm$ sample standard deviation over 5 seeds. Parentheses list training depths selected across seeds, with seed counts as superscripts when more than one depth was selected; executed depth is $\min(K_{\text{train}}, B)$. Ref.: reward of the budget-specific privileged reference. Gap recovery is undefined when the reference does not exceed the fixed baseline and is omitted at $B = 1$, where every method executes one refinement. MiniGrid-DistShift2 is the saturated control. AIC trails on MiniGrid-FourRooms and OGBench puzzle-4x4-play-v0 at B=8.

Task	B	AIC	Fixed (K_{train})	Δ	Ref.	Ref. gap	W/L/T
MiniGrid-DistShift2	1	1.0000±.0000	1.0000±.0000 (1)	.0000	—	—	0/0/5
	2	1.0000±.0000	1.0000±.0000 (1)	.0000	1.0000	—	0/0/5
	4	1.0000±.0000	1.0000±.0000 (1)	.0000	1.0000	—	0/0/5
	8	1.0000±.0000	1.0000±.0000 (1)	.0000	1.0000	—	0/0/5
HalfCheetah-v4	1	.9073±.0008	.9079±.0008 (1)	−.0006	—	—	2/3/0
	2	.9269±.0006	.9270±.0003 (2)	−.0001	.9437	−0.6%	3/2/0
	4	.9331±.0004	.9314±.0002 (4)	+.0017	.9612	5.7%	5/0/0
	8	.9348±.0001	.9329±.0002 (8)	+.0019	.9744	4.6%	5/0/0
GPT-2 on GLUE-RTE	1	.7912±.0027	.7932±.0022 ($1^{(4)}, 4^{(1)}$)	−.0020	—	—	2/2/1
	2	.8230±.0042	.8216±.0020 ($2^{(3)}, 4^{(2)}$)	+.0014	.8852	2.2%	3/2/0
	4	.8298±.0011	.8276±.0042 (4)	+.0022	.9193	2.4%	3/1/1
	8	.8334±.0009	.8311±.0024 (8)	+.0023	.9323	2.3%	4/1/0
OGBench cube-double-play-v0	1	.2453±.0019	.2446±.0031 (1)	+.0007	—	—	3/2/0
	2	.3311±.0036	.3305±.0056 (2)	+.0006	.2555	—	3/2/0
	4	.4047±.0060	.4033±.0047 (4)	+.0014	.3871	—	3/2/0
	8	.4299±.0028	.4281±.0060 (8)	+.0018	.5646	1.3%	4/1/0
OGBench scene-play-v0	1	.2535±.0042	.2538±.0027 (1)	−.0003	—	—	3/2/0
	2	.3216±.0034	.3152±.0036 (2)	+.0064	.3587	14.7%	5/0/0
	4	.3965±.0045	.3957±.0069 (4)	+.0008	.5957	0.4%	2/3/0
	8	.4167±.0048	.4075±.0044 (8)	+.0092	.5565	6.2%	5/0/0
ManiSkill PickCube-v1	1	.4062±.0058	.4132±.0027 (1)	−.0070	—	—	1/4/0
	2	.4756±.0018	.4732±.0051 (2)	+.0024	.5589	2.8%	4/1/0
	4	.4954±.0036	.4911±.0027 (4)	+.0043	.6344	3.0%	5/0/0
	8	.4926±.0045	.4741±.0034 (8)	+.0185	.6756	9.2%	5/0/0
ManiSkill StackCube-v1	1	.3748±.0060	.3746±.0037 (1)	+.0002	—	—	2/3/0
	2	.4251±.0014	.4217±.0031 (2)	+.0034	.4884	5.1%	4/1/0
	4	.4554±.0024	.4508±.0071 (4)	+.0046	.5946	3.2%	3/2/0
	8	.4579±.0036	.4458±.0060 (8)	+.0121	.6274	6.7%	4/0/1
AntMaze (main)	1	.4567±.0024	.4589±.0019 (1)	−.0022	—	—	1/4/0
	2	.5017±.0034	.5040±.0046 (2)	−.0023	.6135	−2.1%	2/3/0
	4	.5208±.0026	.5095±.0052 (4)	+.0113	.7043	5.8%	4/1/0
	8	.5209±.0034	.5018±.0022 ($2^{(1)}, 8^{(4)}$)	+.0191	.7367	8.1%	5/0/0
AntMaze (low-het.)	1	.6881±.0040	.6916±.0021 (1)	−.0035	—	—	1/4/0
	2	.6752±.0056	.6630±.0088 ($1^{(2)}, 2^{(3)}$)	+.0122	.8049	8.6%	5/0/0
	4	.6743±.0053	.6683±.0085 (4)	+.0060	.8618	3.1%	4/1/0
	8	.6776±.0025	.6704±.0047 (8)	+.0072	.8685	3.6%	5/0/0
RareEvent-StackCube-Misalign	1	.6130±.0052	.6338±.0039 (1)	−.0208	—	—	0/5/0
	2	.5831±.0090	.5741±.0063 (2)	+.0090	.7084	6.7%	4/1/0
	4	.5851±.0151	.5632±.0064 (4)	+.0219	.7412	12.3%	5/0/0
	8	.5918±.0050	.5575±.0125 ($1^{(2)}, 8^{(3)}$)	+.0343	.7523	17.6%	5/0/0
MiniGrid-FourRooms	1	.8216±.0148	.7928±.0604 ($1^{(3)}, 4^{(2)}$)	+.0288	—	—	3/2/0
	2	.8162±.0103	.8216±.0040 ($1^{(2)}, 2^{(3)}$)	−.0054	.8154	—	1/2/2
	4	.9171±.0040	.9189±.0000 ($4^{(3)}, 8^{(2)}$)	−.0018	.9153	—	0/1/4
	8	.9514±.0103	.9730±.0064 (8)	−.0216	.9730	—	0/5/0
OGBench puzzle-4x4-play-v0	1	.2067±.0021	.2044±.0035 (1)	+.0023	—	—	4/1/0
	2	.2997±.0033	.2909±.0021 (2)	+.0088	.3163	34.6%	5/0/0
	4	.3670±.0040	.3578±.0061 (4)	+.0092	.4235	14.0%	5/0/0
	8	.4022±.0048	.4069±.0025 (8)	−.0047	.5185	−4.2%	0/5/0

Table 15: **AIC's mean refinement depth at each nominal budget.** $B = 1$ forces one refinement; the $B = 8$ column repeats Table 10.

Task	$B = 1$	$B = 2$	$B = 4$	$B = 8$
MiniGrid-DistShift2	1.00	1.00	1.00	1.00
HalfCheetah-v4	1.00	1.83	2.71	3.04
GPT-2 on GLUE-RTE	1.00	1.74	2.62	3.07
OGBench cube-double-play-v0	1.00	1.92	3.58	5.19
AntMaze (low-het.)	1.00	1.52	2.03	2.24
OGBench scene-play-v0	1.00	1.90	3.49	5.05
ManiSkill StackCube-v1	1.00	1.88	3.12	3.77
ManiSkill PickCube-v1	1.00	1.81	2.93	3.47
AntMaze (main)	1.00	1.79	2.80	3.15
RareEvent-StackCube-Misalign	1.00	1.48	1.71	1.86
MiniGrid-FourRooms	1.00	1.63	2.21	2.43
OGBench puzzle-4x4-play-v0	1.00	1.94	3.66	5.60
Mean (12 tasks)	1.00	1.70	2.66	3.32

Table 16: **Removed-work accounting for the truncation study.** Selected fraction: share of decisions at or above the seed's 75th-percentile depth, including ties. Removed refinements are per intervened decision. Equal-work random: random truncation matched to the targeted set's total removed refinements. Targeted drops are those of Table 3.

	Selected	Removed refinements		Equal-work	Targeted−
Task	fraction	Targeted	Random	random drop	equal-work random
MiniGrid-DistShift2 (sat.)	1.00	0.00	0.00	.0000	.0000
AntMaze (low-het.)	.27	2.90	1.24	−.0041	.0210
HalfCheetah-v4	.29	4.40	2.04	.0219	.0028
GPT-2 on GLUE-RTE	.30	4.30	2.07	.0352	.0149
AntMaze (main)	.28	4.50	2.15	.0661	.0184
ManiSkill PickCube-v1	.35	4.80	2.47	.0843	.0219
ManiSkill StackCube-v1	.30	5.00	2.77	.0951	.0197
OGBench cube-double-play-v0	.31	6.60	4.19	.2118	−.0027
OGBench scene-play-v0	.29	6.50	4.05	.1912	.0033
RareEvent-StackCube-Misalign	.26	2.77	0.86	−.0031	.0065
MiniGrid-FourRooms	.27	3.60	1.43	.1118	.0071
OGBench puzzle-4x4-play-v0	.33	6.80	4.60	.2141	−.0036
mean over 9 positive comparisons				.0776	.0118

Table 17: PickCube phase analysis over 25,000 evaluation decisions. Mean depth is mean ± standard deviation across five seeds; high-depth states satisfy $\tau \geq 5$, the 75th-percentile depth, which selects 35.4% of all decisions because of ties at the cutoff. Deeper-refinement gain is the mean change in normalized return-to-go when a state's executed depth is increased by two refinements; early-exit change is the corresponding change when the state is forced to depth one.

Phase	Share	Mean depth	High-depth states	Deeper-refinement gain	Early-exit change
Approach	56.1%	$1.8 \pm .2$	12%	+.005	−.004
Grasp/contact	14.6%	$5.7 \pm .4$	68%	+.046	−.061
Lift	14.6%	$4.9 \pm .3$	54%	+.038	−.047
Stabilization/recovery	14.7%	$6.2 \pm .5$	74%	+.053	−.069

Table 18: **Memory tasks at $B = 8$.** Fixed uses depth eight on Memory and DoorKey; PickCube and AntMaze repeat the baselines in Table 1. These runs are independent of the stopping-by-supervision factorial. P/R: precision and recall of high-depth decisions at critical states (Section P).

	AIC		ACT-ponder		non-VoC	Fixed	Tgt−Rnd	P/R
Task	Rew.	$\bar{d}$	Rew.	$\bar{d}$	Rew.	Rew.	gap	(crit. states)
MiniGrid *Memory*	.5715	3.31	.5690	3.62	.5677	.5504	+.056	.42/.79
MiniGrid *DoorKey*	.5470	3.36	.5413	3.86	.5431	.5259	+.059	.44/.85
Benchmark tasks at $B = 8$, for scale.								
ManiSkill PickCube-v1	.4926	3.47	.4910	3.80	.4907	.4741	+.068	.49/.74
AntMaze (main)	.5209	3.15	.5200	3.54	.5208	.5018	+.062	.41/.77

Table 19: **Timing and computation.** Inference latency is the median per-decision time at batch size one; training time is wall-clock time for the full run. Mean steps are refinements per decision.

Setting	Fixed-8	AIC	Mean steps (Fixed-8 / AIC)	Change
GPT-2/RTE inference	9.41 ms	8.68 ms	8 / 4.08	-7.8% latency
HalfCheetah inference	0.86 ms	0.48 ms	8 / 3.04	-44% latency
AntMaze inference	0.89 ms	0.53 ms	8 / 3.15	-40% latency
PickCube inference	0.92 ms	0.57 ms	8 / 3.47	-38% latency
StackCube inference	0.94 ms	0.65 ms	8 / 3.77	-31% latency
PickCube training	5.9 h	7.2 h	all eight depths	$1.22\times$ time

Table 20: **Attaching AIC to each policy family.** One refinement is one denoising step; the intermediate action is the one-step clean-action prediction $\hat{a}_0$.

Family	Training regime	Halting update	Critic or baseline	Fixed steps
DiffusionQL	offline	critic-weighted, score function	DiffusionQL double-Q critic	8
IDQL	offline	critic-weighted, score function	IQL Q-function	8
DSAC-D	off-policy	critic-weighted, score function	mean of the distributional critic	8
DPPO	on-policy	joint PPO ratio	GAE with state-only baseline	8
D3P	on-policy	joint PPO ratio	GAE with state-only baseline	8

Table 21: **Full transfer results at nominal $B = 8$.** Fixed and AIC give mean return with the standard deviation over five seeds as a subscript; Gain is $\Delta = R_{\mathrm{AIC}} - R_{\mathrm{Fixed}}$ with its standard error and AIC Steps is the mean halting depth with its standard deviation. Fixed executes eight refinement steps per action in every family.

Base Policy	Task / Env	Fixed	AIC	Gain	AIC Steps
DiffusionQL	HalfCheetah-v4	$.934_{\pm.004}$	$.936_{\pm.004}$	$+.002_{\pm.003}$	$2.8_{\pm.2}$
DiffusionQL	ManiSkill PickCube-v1	$.505_{\pm.005}$	$.525_{\pm.005}$	$+.020_{\pm.003}$	$3.8_{\pm.3}$
DiffusionQL	ManiSkill StackCube-v1	$.486_{\pm.004}$	$.501_{\pm.006}$	$+.015_{\pm.003}$	$4.1_{\pm.3}$
DiffusionQL	OGBench scene-play-v0	$.430_{\pm.005}$	$.442_{\pm.004}$	$+.012_{\pm.003}$	$6.2_{\pm.2}$
DiffusionQL	AntMaze main	$.520_{\pm.004}$	$.537_{\pm.004}$	$+.017_{\pm.003}$	$3.6_{\pm.2}$
DiffusionQL	OGBench puzzle-4x4-play-v0	$.421_{\pm.004}$	$.418_{\pm.005}$	$-.003_{\pm.003}$	$7.2_{\pm.4}$
DiffusionQL	MiniGrid-FourRooms	$.970_{\pm.008}$	$.955_{\pm.009}$	$-.015_{\pm.005}$	$2.6_{\pm.1}$
DiffusionQL	RareEvent-StackCube-Misalign	$.590_{\pm.008}$	$.615_{\pm.009}$	$+.025_{\pm.005}$	$1.8_{\pm.1}$
DPPO	HalfCheetah-v4	$.933_{\pm.003}$	$.934_{\pm.003}$	$+.001_{\pm.002}$	$2.6_{\pm.2}$
DPPO	ManiSkill PickCube-v1	$.492_{\pm.006}$	$.510_{\pm.005}$	$+.018_{\pm.003}$	$3.5_{\pm.2}$
DPPO	ManiSkill StackCube-v1	$.471_{\pm.005}$	$.484_{\pm.005}$	$+.013_{\pm.003}$	$3.8_{\pm.1}$
DPPO	OGBench scene-play-v0	$.417_{\pm.004}$	$.425_{\pm.005}$	$+.008_{\pm.003}$	$5.6_{\pm.2}$
DPPO	AntMaze main	$.510_{\pm.004}$	$.525_{\pm.003}$	$+.015_{\pm.002}$	$3.3_{\pm.3}$
DPPO	OGBench puzzle-4x4-play-v0	$.410_{\pm.004}$	$.405_{\pm.004}$	$-.005_{\pm.003}$	$6.8_{\pm.4}$
DPPO	MiniGrid-FourRooms	$.968_{\pm.008}$	$.950_{\pm.008}$	$-.018_{\pm.005}$	$2.5_{\pm.2}$
DPPO	RareEvent-StackCube-Misalign	$.575_{\pm.009}$	$.597_{\pm.007}$	$+.022_{\pm.005}$	$1.9_{\pm.2}$
D3P	HalfCheetah-v4	$.935_{\pm.004}$	$.937_{\pm.004}$	$+.002_{\pm.003}$	$2.9_{\pm.2}$
D3P	ManiSkill PickCube-v1	$.515_{\pm.005}$	$.538_{\pm.004}$	$+.023_{\pm.003}$	$4.0_{\pm.2}$
D3P	ManiSkill StackCube-v1	$.496_{\pm.005}$	$.514_{\pm.005}$	$+.018_{\pm.003}$	$4.3_{\pm.3}$
D3P	OGBench scene-play-v0	$.438_{\pm.004}$	$.452_{\pm.005}$	$+.014_{\pm.003}$	$6.5_{\pm.3}$
D3P	AntMaze main	$.525_{\pm.004}$	$.543_{\pm.004}$	$+.018_{\pm.003}$	$3.8_{\pm.3}$
D3P	OGBench puzzle-4x4-play-v0	$.420_{\pm.004}$	$.422_{\pm.004}$	$+.002_{\pm.003}$	$7.4_{\pm.2}$
D3P	MiniGrid-FourRooms	$.969_{\pm.007}$	$.957_{\pm.007}$	$-.012_{\pm.004}$	$2.8_{\pm.3}$
D3P	RareEvent-StackCube-Misalign	$.600_{\pm.008}$	$.630_{\pm.009}$	$+.030_{\pm.005}$	$2.0_{\pm.1}$
DSAC-D	HalfCheetah-v4	$.934_{\pm.004}$	$.935_{\pm.004}$	$+.001_{\pm.003}$	$2.2_{\pm.1}$
DSAC-D	ManiSkill PickCube-v1	$.485_{\pm.006}$	$.495_{\pm.006}$	$+.010_{\pm.004}$	$3.1_{\pm.3}$
DSAC-D	ManiSkill StackCube-v1	$.462_{\pm.005}$	$.470_{\pm.005}$	$+.008_{\pm.003}$	$3.4_{\pm.1}$
DSAC-D	OGBench scene-play-v0	$.405_{\pm.005}$	$.409_{\pm.005}$	$+.004_{\pm.003}$	$5.0_{\pm.2}$
DSAC-D	AntMaze main	$.500_{\pm.004}$	$.510_{\pm.004}$	$+.010_{\pm.003}$	$3.0_{\pm.2}$
DSAC-D	OGBench puzzle-4x4-play-v0	$.407_{\pm.003}$	$.401_{\pm.003}$	$-.006_{\pm.002}$	$6.5_{\pm.2}$
DSAC-D	MiniGrid-FourRooms	$.970_{\pm.006}$	$.950_{\pm.008}$	$-.020_{\pm.004}$	$2.3_{\pm.2}$
DSAC-D	RareEvent-StackCube-Misalign	$.560_{\pm.009}$	$.572_{\pm.008}$	$+.012_{\pm.005}$	$1.6_{\pm.2}$
IDQL	HalfCheetah-v4	$.933_{\pm.003}$	$.934_{\pm.003}$	$+.001_{\pm.002}$	$2.4_{\pm.1}$
IDQL	ManiSkill PickCube-v1	$.490_{\pm.004}$	$.502_{\pm.006}$	$+.012_{\pm.003}$	$3.3_{\pm.2}$
IDQL	ManiSkill StackCube-v1	$.468_{\pm.005}$	$.478_{\pm.005}$	$+.010_{\pm.003}$	$3.6_{\pm.3}$
IDQL	OGBench scene-play-v0	$.424_{\pm.004}$	$.434_{\pm.005}$	$+.010_{\pm.003}$	$6.0_{\pm.3}$
IDQL	AntMaze main	$.506_{\pm.004}$	$.520_{\pm.004}$	$+.014_{\pm.003}$	$3.4_{\pm.3}$
IDQL	OGBench puzzle-4x4-play-v0	$.413_{\pm.004}$	$.414_{\pm.004}$	$+.001_{\pm.003}$	$7.0_{\pm.4}$
IDQL	MiniGrid-FourRooms	$.969_{\pm.006}$	$.955_{\pm.008}$	$-.014_{\pm.004}$	$2.5_{\pm.2}$
IDQL	RareEvent-StackCube-Misalign	$.570_{\pm.008}$	$.588_{\pm.008}$	$+.018_{\pm.005}$	$1.7_{\pm.2}$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the joint RL formulation, conditional allocation result, and scope of the evaluation. Sections 3, 4.2 and 5 explain what the performance comparisons establish, when allocation accounts for gains, and how training cost differs from inference cost.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section O explains the limits of shared value targets, non-monotone refinement, and interventions that match state count but remove different amounts of computation. It also discusses reward-scale sensitivity, training cost, encoder overhead, and the privileged reference.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section B proves the fixed-distribution allocation bound for randomized comparators and states the occupancy conditions for sequential returns. Section A proves the stopping-law and shared-target identities. Section C gives complete proofs of the stopping recursion, first-crossing condition, optimal-distribution characterization, and perturbation bound, with counterexamples and explicit scope. No joint-PPO convergence guarantee is claimed.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architecture, nominal settings and reported in github.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The Project page link added in abstract contains code.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Default optimizer, seeds and loss weights are specified in Section I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The per-budget table reports means and standard deviations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section Q reports relative training and inference timing.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work involves standard reinforcement-learning experiments on public, simulated benchmarks. No human subjects, personal data, or scraped content are used. Authors have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Adaptive stopping can reduce inference work in suitable regimes, but measured latency depends on encoder cost and training is more expensive. These limits are reported explicitly rather than inferring energy savings from refinement counts alone.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper releases small RL policies trained on simulated benchmarks; these do not pose dual-use or misuse risks comparable to large language models and image generators datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All benchmarks and pretrained components are cited at first use: MuJoCo [Todorov et al., 2012] and Gymnasium [Brockman et al., 2016] (Apache-2.0/MIT); ManiSkill2 [Gu et al., 2023] (Apache-2.0); D4RL [Fu et al., 2020] (Apache-2.0); GLUE [Wang et al., 2018a] (research use); GPT-2 [Radford et al., 2019] (Modified MIT);

MiniGrid [Chevalier-Boisvert et al., 2018] (Apache-2.0); OGBench [Park et al., 2025] (MIT); base policies DiffusionQL [Wang et al., 2022], DPPO [Ren et al., 2025], D3P [Yu et al., 2025], DSAC-D [Liu et al., 2025], IDQL [Hansen-Estruch et al., 2023] as cited. We use each within its license terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: AIC code, training scripts, and the targeted-vs-random ablation harness are present on github link given in project page.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The AIC architecture and allocation evaluation do not introduce an LLM-based method. GPT-2 [Radford et al., 2019] serves as a frozen text encoder for the single GPT-2 on GLUE-RTE task, GLUE-RTE. Its role and the refinement applied after encoding are specified in Section D.1.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.