CARE: CERTIFYING ACCELERATION FOR VISION-LANGUAGE-ACTION INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

While vision-language-action (VLA) models have advanced rapidly, running them at every control step remains expensive. Prior work accelerates VLA inference using techniques like action chunking and visual-token pruning, typically evaluating based on latency and average task success. However, acceleration may discard information and break tasks the original policy would solve, a risk hidden by average metrics. Measuring these failures is challenging because action deviations compound over closed-loop trajectories, meaning task failure is only observable across full episodes. We therefore define an acceleration-induced failure via paired rollouts from identical initial conditions, tracking when the reference succeeds but the accelerated policy fails. To manage this, we introduce CARE, an approach for certified accelerator selection. CARE uses paired rollouts on a calibration set to provide finite-sample guarantees that acceleration-induced failure risk stays below a user-specified budget. It deploys the fastest certified candidate, falling back to the reference if none qualify. By relying only on terminal outcomes and measured compute, CARE applies unchanged across diverse acceleration mechanisms, while sequential testing and failure-triggered reference rollouts keep certification affordable. On four LIBERO suites with OpenVLA-OFT, CARE certifies 9.0–$10.8\times$ speedups while guaranteeing (at 95% confidence) that at least 85.8% of reference-solved episodes are preserved. Under tight budgets, selectors without guarantees exceed the budget in up to 75% of trials, whereas CARE stays within budget and its sequential form uses 78.9% fewer rollouts than exhaustive evaluation. CARE further generalizes to flow-step reduction for $\pi_{0.5}$, and to Qwen3.5-9B and Llama-3.1-8B agents in Crafter.

1 INTRODUCTION

Vision-language-action (VLA) models have rapidly advanced multimodal decision-making by mapping visual observations and language instructions to actions (Kim et al., 2024; Black et al., 2024). Their inference cost, however, remains a practical bottleneck: repeatedly running a large model in a closed-loop controller can make execution slow. A growing literature therefore reduces VLA inference cost through action chunking (Kim et al., 2025), visual-token pruning (Chen et al., 2024; Liu et al., 2025), feature caching (Xu et al., 2026), dynamic inference (Yue et al., 2024), and fewer refinement steps for flow-matching policies (Black et al., 2024; Intelligence et al., 2025). These approaches differ substantially in mechanism, but are typically evaluated using the same two quantities: inference cost and average task success.

These metrics describe how fast a policy is and how often it succeeds overall, but not what acceleration breaks. Acceleration may discard visual information, reuse stale features, or reduce re-planning frequency. A faster policy can therefore fail on tasks that the original policy would have solved. Average success does not isolate these regressions: failures shared by both policies count against the accelerated policy, while tasks newly solved by the accelerated policy can compensate for tasks that acceleration destroys. Two policies can consequently have similar average success while behaving very differently. This raises a key deployment question we ask: *how much can we accelerate a policy while bounding how often acceleration breaks tasks the reference already solves?*

We formalize this question through *acceleration-induced failure*: from the same task and initial scene, the reference policy, run at full compute, succeeds while the accelerated policy fails. This event is defined over two complete closed-loop rollouts. Once acceleration changes an action, the next state changes as well, and every subsequent observation and decision may diverge from the

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

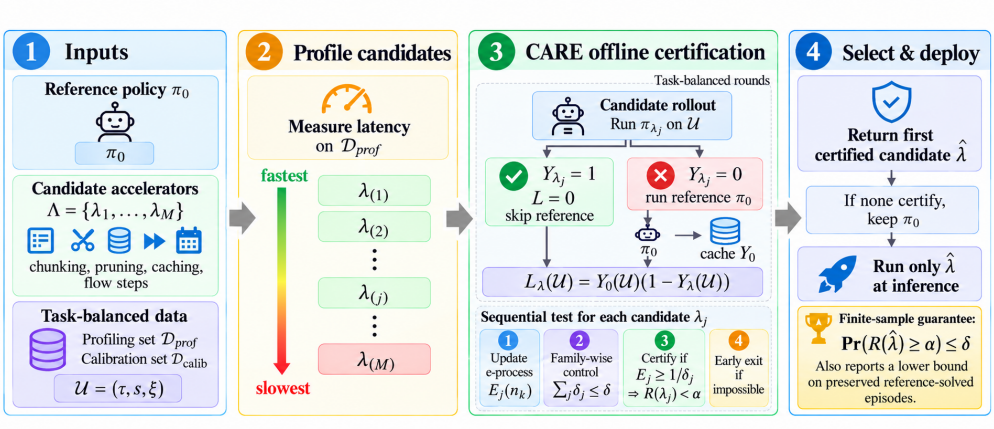

Figure 1: **Overview of CARE.** Given a reference policy π_0 and accelerated candidates $\Lambda = \{\lambda_1, \ldots, \lambda_M\}$, CARE first profiles candidate latency on a disjoint profiling set and orders candidates from fastest to slowest. On a separate calibration set, candidates are evaluated sequentially in task-balanced rounds, with each round containing the same number of episodes from every task. For each paired calibration episode, the reference and accelerated policy start from the same evaluation unit U, and CARE records the episode-level loss $L_\lambda(U) = Y_0(U)\big(1 - Y_\lambda(U)\big)$, which equals one only when the reference succeeds and the accelerated policy fails. An anytime-valid test with family-wise error control certifies candidates while allowing early stopping. To reduce calibration cost, the reference policy is executed only when a candidate fails, and its outcome is cached for reuse. CARE returns the first, and therefore fastest, certified candidate; if none can be certified, it retains π_0. The returned policy satisfies the finite-sample deployment guarantee without adding overhead at inference.

reference trajectory. We show that the resulting episode-level risk cannot in general be recovered from the two policies' marginal success rates or from per-decision measurements such as confidence or action coverage (Appendix A.5), so it must be measured on paired, complete episodes. This makes certification challenging in two ways. Statistically, the risk must be bounded from a finite number of episodes, and choosing among many candidate accelerators creates many chances to accept an unsafe one by luck. Computationally, every evaluation requires closed-loop rollouts, including rollouts of the expensive reference policy that acceleration is meant to avoid.

We introduce CARE, an approach for certified accelerator selection. CARE runs once, offline, before deployment. Given a reference policy, a family of acceleration configurations, and a user-defined risk budget, it executes paired rollouts on a calibration set, records whether each rollout succeeds, and constructs a finite-sample certificate for each candidate's acceleration-induced failure risk. Concretely, it leverages Learn-then-Test (Angelopoulos et al., 2025) with finite-sample concentration bounds (Hoeffding, 1963; Bentkus, 2004) to test whether each candidate's risk is below the budget while controlling errors across the candidate family, and returns the fastest certified candidate, or retains the reference when the data do not support any acceleration. At deployment, only the selected policy runs. CARE is also model and accelerator-agnostic: it treats each candidate as a black-box policy and requires only terminal outcomes and measured compute, so the same procedure applies to action chunking, token pruning, feature caching, and their combinations.

Beyond this formulation, CARE also makes certification itself adaptive and computationally efficient, as illustrated in Figure 1. The candidate family is ordered by latency measured on a separate profiling set, and CARE evaluates candidates from fastest to slowest, stopping at the first one it certifies, so slower candidates need not be evaluated at all. For each candidate, it applies anytime-valid sequential tests on task-balanced rounds of scenes, stopping as soon as the evidence certifies the candidate or rules it out, with a guarantee that remains valid even though tasks differ widely in risk. It further avoids unnecessary reference executions: if a candidate succeeds on a scene, it cannot have caused an acceleration-induced failure there, so the reference is run only when a candidate fails, and its outcome is cached for later candidates. These mechanisms preserve the finite-sample guarantee while substantially reducing the rollout cost of certification.

We evaluate CARE with OpenVLA-OFT (Kim et al., 2025) on four LIBERO (Liu et al., 2023) suites, using candidates that combine action chunking with visual-token pruning and feature caching. CARE

achieves 9.0–$10.8\times$ speedups over one-step replanning while guaranteeing, simultaneously at 95% confidence, that the selected policies preserve at least 85.8% of reference-solved episodes. Under tight risk budgets, common selectors, such as choosing the fastest candidate, matching average success, or tuning on validation data, exceed the budget in 13–75% of resampled trials, whereas CARE stays within budget in every trial by retaining the reference when the evidence is insufficient. Sequential fastest-first testing with failure-triggered reference evaluation reduces certification from $9{,}000$ to $1{,}997$ rollouts (77.8% fewer) and from 38.6 to 7.0 hours, running the reference only 47 instead of $1{,}000$ times. On a larger family of 24 configurations per suite, CARE uses 78.9% fewer rollouts than exhaustive evaluation and is the only compared method that certifies acceleration on every suite. The same method certifies reduced flow steps for $\pi_{0.5}$ (Intelligence et al., 2025) on LIBERO suites, and compute-reduced Qwen3.5-9B (Yang et al., 2025) and Llama-3.1-8B (Grattafiori et al., 2024) agents in Crafter (Hafner, 2021). In summary, our key contributions are:

- We propose CARE, a certify-then-select approach that returns the fastest candidate whose acceleration-induced failure risk is certified below a user-specified budget, with a finite-sample guarantee that holds across the candidate family and a lower bound on the fraction of reference-solved episodes preserved.
- We make certification efficient through two designs tailored to paired, episode-level evaluation: task-balanced sequential testing that stops adaptively and stays valid across tasks of different risk, and failure-triggered reference evaluation that runs the reference only when a candidate fails, without changing any decision.
- Across OpenVLA-OFT and $\pi_{0.5}$ on LIBERO and language-model agents in Crafter, CARE certifies large speedups where common selectors violate the budget, while reducing certification cost by nearly 80%.

2 Related Work

Efficient inference and adaptive computation. Prior work reduces the inference cost of multimodal policies by changing either what is computed or how often computation occurs. VLA acceleration includes action chunking (Kim et al., 2025), visual-token pruning (Chen et al., 2024; Liu et al., 2025; Wang et al., 2025; Pei et al., 2026), feature caching (Xu et al., 2026), and early-exit or dynamic-inference methods (Schuster et al., 2022; Yue et al., 2024). For flow-matching policies such as π_0 and $\pi_{0.5}$, reducing the number of integration steps provides another compute knob (Black et al., 2024; Intelligence et al., 2025). Related methods also reduce language-model reasoning through dynamic pruning or stopping (Zheng et al., 2026). These approaches primarily optimize computation while preserving aggregate performance. They are evaluated by latency and average task success, which reveal neither which tasks the faster policy newly fails nor whether its success will hold on new scenes.

CARE instead determines whether an accelerated policy has sufficient evidence for deployment under a specified acceleration-induced failure budget. Using only paired terminal outcomes and measured inference cost, it is model agnostic and applies the same certification procedure to action chunking, token pruning, feature caching, flow-step reduction, and their combinations. CARE thereby identifies the fastest candidate that can be certified for deployment under a risk budget.

Finite-sample risk control. Learn-then-Test (LTT) converts candidate selection into multiple hypothesis testing over a finite family (Angelopoulos et al., 2025; Bates et al., 2021), while Pareto Testing uses an estimated candidate ordering to improve selection efficiency (Laufer-Goldshtein et al., 2022). Conformal risk control provides related guarantees under additional loss structure (Angelopoulos et al., 2024; Angelopoulos, 2026). Anytime-valid inference, e-values, and test supermartingales permit repeated inspection and early stopping without invalidating the guarantee (Ville, 1939; Vovk & Wang, 2021; Ramdas et al., 2023; Howard et al., 2021). These tools have been applied to early exits and adaptive computation (Schuster et al., 2022; Jazbec et al., 2024). Safe policy improvement instead compares expected return with a baseline (Thomas et al., 2015), while negative-flip analyses examine inputs on which an updated model fails after the original succeeds (Yan et al., 2021).

Unlike prior work on per-prediction consistency or negative flips over fixed inputs, CARE targets closed-loop acceleration risk: the probability that an accelerated policy fails on an episode the reference would solve. Because acceleration changes actions and therefore future states, this risk cannot in general be inferred from marginal success rates, average-success differences, or per-step guarantees; it must be certified on paired episodes. To make this certification computationally

tractable while preserving safety guarantees, CARE introduces two novel mechanisms: *task-balanced sequential certification* to preserve anytime validity under heterogeneous task risks, and *failure-triggered reference execution*, which avoids running the reference whenever the accelerated policy already succeeds. Together, these mechanisms retain the deployment guarantee while reducing the cost of closed-loop certification.

3 APPROACH

We present CARE, a training-free approach that decides which accelerator may safely replace a reference policy. CARE starts from a computationally heavy reference policy π_0 and a family of M candidate acceleration configurations $\Lambda = \{\lambda_1, \ldots, \lambda_M\}$. Each configuration λ, such as an action-chunk length, a visual-token pruning ratio, or a number of flow-integration steps, turns π_0 into a faster policy π_λ. CARE identifies the fastest candidate it can certify as safe on a calibration set, falling back to π_0 if no candidate qualifies.

3.1 PROBLEM FORMULATION

Let an evaluation unit $U = (\tau, s, \xi)$ consist of a task index $\tau \in \{1, \ldots, T\}$, an initial scene s of task τ, and the random seeds ξ used during execution. Running the reference and a candidate on the same unit, that is, from the same scene with the same seeds, is a paired evaluation, and yields binary outcomes $Y_0(U), Y_\lambda(U) \in \{0, 1\}$, with 1 denoting success. Pairing ensures that a difference between the two outcomes reflects the policies rather than different starting conditions. Units are drawn independently and divided into disjoint sets: a calibration set $\mathcal{D}_{\text{calib}}$ used to certify and select, and a test set $\mathcal{D}_{\text{test}}$ used to evaluate the selected policy.

To evaluate safety, we must isolate failures caused specifically by the acceleration method. Standard marginal metrics (comparing overall success rates) fail to do this: they cannot distinguish an acceleration-induced failure from a task that was simply too difficult for the reference policy to solve, and they let tasks the accelerated policy happens to repair cancel tasks it breaks. Nor can such failures be measured from individual decisions, since a cheaper decision changes every state that follows (Appendix A.5). Therefore, we define the acceleration-induced failure (AIF) loss as $L_\lambda(U) = Y_0(U)\big(1 - Y_\lambda(U)\big) \in \{0, 1\}$, which equals 1 if and only if acceleration degrades a reference success into a failure. Deployment risk is then defined under an equal-weight task mixture, so that every task contributes equally: $R(\lambda) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[L_\lambda(U) \mid \tau = t]$.

Let λ_0 denote the reference itself, so $R(\lambda_0) = 0$. Given a risk budget $\alpha \in (0, 1)$, the largest acceptable AIF risk, and an error level $\delta \in (0, 1)$, we seek a rule that maps $\mathcal{D}_{\text{calib}}$ to a configuration $\widehat{\lambda} \in \Lambda \cup \{\lambda_0\}$, is as fast as possible, and satisfies the PAC-style guarantee (Valiant, 2026):

$$\Pr\big(R(\widehat{\lambda}) \geq \alpha\big) \leq \delta, \tag{1}$$

where the probability is over the calibration set and its executions.

We use $R(\lambda)$ as the deployment constraint because it directly measures the probability that acceleration causes a regression on a random deployment episode. We also report the retention $\mathrm{Ret}(\lambda) = \Pr(Y_\lambda{=}1 \mid Y_0{=}1) = 1 - R(\lambda)/\Pr(Y_0{=}1)$, the fraction of reference-solved episodes that acceleration preserves, with expectations under the same task mixture. Hence $R(\lambda) < \alpha$ implies $\mathrm{Ret}(\lambda) > 1 - \alpha/\Pr(Y_0{=}1)$: the two criteria agree when the reference is reliable but can diverge when it often fails. CARE reports a simultaneous lower confidence bound on $\mathrm{Ret}(\widehat{\lambda})$ and, when retention is the requirement, can certify a retention budget directly (Appendix A.6). The two summaries answer complementary questions: AIF risk controls the deployment-wide probability of regression, while retention reports how much of the reference's solved set is preserved.

3.2 CERTIFY THEN SELECT

To achieve the objective, we formulate candidate selection as a statistical hypothesis testing problem following the Learn-then-Test framework (Angelopoulos et al., 2025). For each candidate λ_j, we test the null hypothesis that it is unsafe: $H_j : R(\lambda_j) \geq \alpha$. Candidate λ_j is certified for deployment only if H_j is rejected at significance level δ_j.

Family-wise error rate control. Testing multiple candidates inflates the probability of a false positive, i.e., certifying an unsafe candidate by sheer sampling chance. To prevent this, we bound the family-wise error rate (FWER) by allocating individual error budgets such that $\sum_{j=1}^{M} \delta_j \leq \delta$, e.g.,

$\delta_j = \delta/M$. By the union bound, the overall probability of falsely certifying *any* unsafe candidate across the entire pool remains bounded by δ. Outside this event every certified candidate is safe, so deploying the fastest certified candidate, or λ_0 if none is certified, satisfies Eq. 1.

Fixed-sample p-values. Suppose we evaluate a candidate on a fixed sample of n calibration units. We write $L_{j,i} = L_{\lambda_j}(U_i)$ and $\widehat{R}_j = \frac{1}{n}\sum_{i=1}^{n} L_{j,i}$. A small $\widehat{R}_j$ is evidence against H_j, and a p-value $\widehat{p}_j$ quantifies it; $\widehat{p}_j$ is *valid* if $\Pr(\widehat{p}_j \le s) \le s$ for all $s \in [0,1]$ whenever H_j holds. We use the Hoeffding–Bentkus p value (Bates et al., 2021; Bentkus, 2004), which is valid for $[0,1]$-valued losses at every sample size without distributional assumptions (Eq. 3, Appendix A.2), and certify λ_j if $\widehat{p}_j \le \delta_j$.

3.3 Task-Balanced Sequential Certification

While the fixed-sample test is statistically valid, evaluating every candidate over the entire calibration set is computationally wasteful. We instead want to halt evaluation the moment a candidate gathers enough evidence to be certified, or accumulates too many failures to ever recover. To this end, we extend CARE into an adaptive sequential certification procedure (Algorithm 1, Appendix A.1).

Task-balanced rounds. Benchmarks provide scenes task by task, and tasks differ widely in risk. If units arrived in task order, an early inspection could see only low-risk tasks and certify a candidate that is unsafe on average. We therefore group calibration units into rounds of m units per task, $B = mT$ units in total, and inspect the evidence only at round boundaries $n_k = kB$, $k = 1, \dots, K$, so every inspection reflects all tasks in their target proportions.

Anytime-valid evidence. Recomputing $\widehat{p}_j$ at every inspection would inflate the error, since each inspection is another chance for a lucky fluctuation. We instead use an e-process, whose validity does not depend on when evaluation stops. With $f_j(n_k) = \sum_{i=1}^{n_k} L_{j,i}$ the number of AIF losses among the first n_k units,

$$E_j(n_k) = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \left(\frac{q}{\alpha}\right)^{f_j(n_k)} \left(\frac{1-q}{1-\alpha}\right)^{n_k - f_j(n_k)}, \qquad \mathcal{Q} \subset [0,\alpha). \tag{2}$$

Each term is a likelihood ratio comparing the observed losses under a safe risk $q < \alpha$ against the boundary risk α; it grows when losses are rarer than α predicts. Since the true risk is unknown, we average over a finite grid $\mathcal{Q}$ of safe values. Under H_j, task-balanced rounds make E_j a non-negative supermartingale across inspections, i.e., it does not grow in expectation, even though tasks differ in risk (Lemma 2). By Ville's inequality (Ville, 1939), the probability that $E_j(n_k)$ ever reaches $1/\delta_j$ at any inspection is bounded by δ_j. Thus, candidate λ_j is safely certified the moment $E_j(n_k) \ge 1/\delta_j$, even though this moment depends on the data.

Fastest-first search & early exit. To find the fastest safe controller, candidates are pre-sorted from fastest to slowest using the disjoint, small profiling set $\mathcal{D}_{\text{prof}}$ dedicated solely to measuring latency. Algorithm 1 halts and deploys the first certified candidate it finds. Conversely, if a candidate's early losses are so severe that $E_j(n_K) < 1/\delta_j$ even if zero future losses were to occur ($L_{j,i} = 0$ for $i > n_k$), the candidate can never be certified, and an early exit immediately abandons it to save compute (Appendix A.3).

3.4 Failure-Triggered Reference Evaluation

While sequential testing reduces the number of calibration units each candidate needs, executing the heavy reference policy π_0 during calibration remains a computational bottleneck. CARE mitigates this via a failure-triggered mechanism.

Exploiting the structure of the AIF loss $L_{j,i} = Y_0(U_i)\big(1 - Y_j(U_i)\big)$, we observe that if candidate λ_j succeeds ($Y_j(U_i) = 1$), the loss is $L_{j,i} = 0$ regardless of $Y_0(U_i)$, rendering the reference outcome irrelevant. Therefore, the reference π_0 is executed only when the candidate fails ($Y_j(U_i) = 0$). Furthermore, reference outcomes $Y_0(U_i)$ are stored in a cache $\mathcal{C}$ shared across candidates, so a later failure on the same unit reuses the cached outcome. As a result, π_0 is executed at most once per unit, and only on units where some evaluated candidate fails, bounding the total number of reference executions by $N_{\text{ref}} = |\mathcal{C}| \le \min\{n_K, \sum_j F_j\}$, where F_j is the number of failures of λ_j, while yielding exactly the same losses, and hence the same certification decisions, as exhaustive paired evaluation (Lemma 1, Appendix A.3).

Combining fastest-first sequential certification with failure-triggered reference evaluation gives the complete CARE controller. Candidate configurations are considered in increasing order of measured inference cost, evaluated on task-balanced rounds, and certified with anytime-valid evidence. The accelerated policy induced by the first certified configuration is deployed; if no candidate can be certified, the reference policy π_0 is retained. The candidate ordering is fixed in advance on $\mathcal{D}_{\text{prof}}$, and early exit and failure-triggered reference evaluation affect only calibration cost, not which candidate is certified, so the finite-sample guarantee in Eq. 1 holds for the deployed configuration (Theorem 1).

4 EXPERIMENTS

We evaluate CARE across different benchmarks, LIBERO (Liu et al., 2023) and Crafter (Hafner, 2021), spanning multiple model families, including OpenVLA-OFT (Kim et al., 2025), the flow-matching policy $\pi_{0.5}$ (Intelligence et al., 2025), and the language-model agents Qwen3.5-9B (Yang et al., 2025) and Llama-3.1-8B (Grattafiori et al., 2024). Our experiments address four research questions: **Q1:** Does CARE select useful accelerated controllers while maintaining family-level AIF guarantees? **Q2:** How does CARE compare against baseline selectors in maintaining safe risk control? **Q3:** Do sequential and reference-query rules effectively reduce certification cost? **Q4:** Does CARE generalize across diverse model architectures and tasks?

4.1 CERTIFIED OPENVLA DEPLOYMENT

We first evaluate OpenVLA-OFT on the four LIBERO suites (Figure 2, Appendix C), Goal, Spatial, Object, and Long, each with ten tasks, using one-step replanning (H1) as the reference. The candidate family contains eight configurations, all with action horizon 8: plain execution, FastV-25, FastV-50, FastV-75 (Chen et al., 2024), VLA-Cache (Xu et al., 2026), VLA-Pruner (Liu et al., 2025), SpecPrune (Wang et al., 2025), and VLA-ADP (Pei et al., 2026). We evaluate 500 matched scenes per suite, for 2,000 initial scenes and 18,000 rollouts across the reference and candidates. Certification uses 250 task-balanced episodes per suite; the remaining 250 episodes form the test set.

For LIBERO, each evaluation unit fixes the task, initial scene, and random seed before either policy is executed. We set the AIF risk budget to $\alpha = 5.0\%$ and the error probability to $\delta = 5.0\%$; see a sensitivity analysis in Appendix B.2. On the test set, we report two metrics: (1) speedup over reference, the ratio of the reference policy's mean inference time to that of the deployed controller (larger is faster); and (2) preserved success, a family-corrected 95% lower confidence bound on the fraction of reference-solved episodes preserved under acceleration. The family-wise correction keeps the reported bounds valid after selecting among multiple candidates across all four suites.

Table 1: **Certified CARE deployments.** Speedup is relative to H1; preserved is the simultaneous 95% lower bound on the fraction of H1-solved episodes retained.

Suite	Selected policy	Speedup	Preserved
Goal	FastV-25 H8	$9.02\times$	93.5%
Spatial	FastV-75 H8	$10.08\times$	95.7%
Object	Plain H8	$9.56\times$	92.2%
Long	VLA-Cache H8	$10.81\times$	85.8%

Table 1 summarizes the certification outcomes. CARE certifies an accelerated policy on every suite, achieving 9.02–$10.81\times$ speedups, while the simultaneous lower bounds guarantee that the selected policies preserve 85.8%–95.7% of reference-solved episodes. These results answer **Q1**: CARE selects useful accelerated policies with family-level guarantees.

4.2 COMPARISON WITH DEPLOYMENT SELECTORS

We compare CARE with alternative selectors through controlled resampling. For each LIBERO suite, the 500 stored paired episodes define an empirical population. In each of 10,000 trials, we sample 250 calibration episodes, apply each method, and evaluate the selected policy against the full population. We organize the comparison around two questions. First, under tight risk budgets, do commonly used selectors respect the requested risk? Second, at the standard budget, when all risk-controlled methods are valid, which method selects faster policies and requires fewer evaluation rollouts?

Selection validity under tighter budgets. We first examine whether each method respects the requested risk budget. Fixed-n CARE evaluates every candidate on all 250 calibration episodes and applies a family-corrected certificate. Latency only always selects the fastest policy. Average success selects the fastest policy whose empirical success is within one percentage point of H1. Validation tuned selects the fastest policy whose observed AIF rate is below α, without correcting for selection

over multiple candidates. The unpaired test compares marginal success rates rather than paired AIF outcomes. We report the over-budget rate, the percentage of trials in which the selected policy's population AIF exceeds α.

As shown in Table 2, CARE maintains validity under tighter budgets by safely retaining H1 when no accelerated policy has sufficient evidence for certification; its validity thus reflects safe fallback rather than forced acceleration. In contrast, latency-only and average-success selection exceed the budget in 75.0% and 25.0% of trials at $\alpha = 2.0\%$ and 2.5%, respectively, and validation tuning and the unpaired test also frequently violate it. Their apparent speedups arise because they almost always deploy an accelerated policy despite insufficient evidence. These results directly answer **Q2**, confirming that CARE delivers safe risk control where common selectors do not.

Deployment speed and evaluation cost. We next compare selectors at the standard budget $\alpha = 5.0\%$, reporting mean speedup relative to H1, fallback rate, and mean evaluation rollouts. Pareto Testing splits episodes to rank candidates by estimated risk before testing, while cost-order LTT tests policies from fastest to slowest; both ordered methods stop at the first uncertifiable policy. Safe policy improvement controls net success relative to H1, allowing newly solved episodes to offset failures, and therefore does not directly control AIF. Sequential CARE tests paired AIF with interim evidence checks, stops early when certification is achieved or unattainable, and runs H1 only after an accelerated policy fails.

Table 2: **Over-budget rate (%) under tighter AIF budgets.** CARE has no observed over-budget deployments, whereas the other selectors exceed the budget in 13.0–75.0% of trials.

Method	$\alpha = 2.0\%$	$\alpha = 2.5\%$
Latency only	75.0	25.0
Average success	75.0	25.0
Validation tuned	36.8	13.0
Unpaired test	67.1	19.5
CARE	**0.0**	**0.0**

As shown in Table 3, fixed-n CARE achieves the highest mean speedup among risk-controlling selectors ($8.88\times$ with 11.7% fallback). In contrast, Pareto Testing ($6.16\times$, 42.2% fallback) and cost-order LTT ($6.35\times$, 45.5% fallback) fall back more often because their rigid evaluation order can halt before reaching a certifiable policy, and safe policy improvement yields $5.30\times$ with 60.5% fallback. Crucially, sequential CARE maintains a competitive $7.32\times$ speedup (26.9% fallback) while reducing evaluation cost from 2,250 to just 592 rollouts. These results directly answer **Q3**, confirming that sequential and reference-query rules substantially reduce certification cost.

Table 3: **Deployment speed and evaluation cost.** Fixed-n CARE achieves the highest mean speedup; sequential CARE reduces evaluation cost by 73.7% while keeping a $7.32\times$ speedup and less fallback than all baselines.

Method	Mean speedup ($\times$)	Fallback (%)	Mean rollouts
CARE, fixed-n	**8.88**	11.7	2,250
Pareto Testing	6.16	42.2	2,250
Cost-order fixed-sequence LTT	6.35	45.5	2,250
Safe policy improvement	5.30	60.5	2,250
CARE, sequential	7.32	26.9	**592**

4.3 Evaluation with a Broader Candidate Family

We expand the candidate family to 24 OpenVLA-OFT configurations formed by four action horizons, $\mathrm{H} \in \{2, 4, 6, 8\}$, and six inference modes: plain inference, FastV-25, FastV-50, FastV-75, VLA-Cache, and VLA-Pruner. We use fresh calibration streams while retaining the same H1 reference, AIF budget, and family-level risk control. This experiment tests whether CARE remains effective and evaluation-efficient when certification must search a larger and more heterogeneous policy family.

Baselines and metrics. Full-sample Hoeffding–Bentkus (HB) evaluates all 24 candidates and H1 on all 250 calibration episodes. Sequential HB and sequential exact inspect the evidence at five pre-specified points, using Hoeffding–Bentkus and exact-binomial tests, respectively. The plug-in e-process is an alternative anytime-valid sequential test. CARE combines task-balanced sequential testing with fastest-first evaluation, early abandonment of candidates that can no longer certify, and failure-triggered H1 evaluation. For each method, we report the selected acceleration mechanism and horizon per suite, and measure evaluation cost by candidate rollouts, H1 rollouts, their total, and the reduction relative to full-sample HB.

Table 4: **Evaluation with a broader acceleration candidate family.** Reference denotes fallback to H1. CARE is the only method that certifies an accelerated policy on every suite, using 5,263 rollouts, 78.9% fewer than full-sample HB and fewer than every sequential baseline. Rollouts are aggregated over the four LIBERO suites.

Method	Suite	Acceleration	Horizon	Candidate rollouts	Reference rollouts	Total rollouts	Rollout saving (%)
Full-sample HB	Goal	FastV-50	H8	24,000	1,000	25,000	0.0
	Spatial	VLA-Cache	H2				
	Object	VLA-Pruner	H8				
	Long	Reference	–				
Sequential HB	Goal	Reference	–	7,150	156	7,306	70.8
	Spatial	Reference	–				
	Object	VLA-Pruner	H8				
	Long	Reference	–				
Sequential exact	Goal	FastV-50	H8	6,350	135	6,485	74.1
	Spatial	VLA-Cache	H2				
	Object	VLA-Pruner	H8				
	Long	Reference	–				
Plug-in e-process	Goal	FastV-25	H8	5,750	123	5,873	76.5
	Spatial	Reference	–				
	Object	VLA-Pruner	H8				
	Long	Reference	–				
CARE	Goal	**FastV-25**	**H8**	**5,150**	**113**	**5,263**	**78.9**
	Spatial	**FastV-75**	**H8**				
	Object	**Plain**	**H8**				
	Long	**VLA-Cache**	**H8**				

Table 4 shows that expanding the candidate family does not diminish the effectiveness or efficiency of CARE, which certifies an accelerated policy on all four suites. In contrast, full-sample HB and sequential exact retain H1 on Long, the plug-in e-process retains H1 on Spatial and Long, and sequential HB certifies acceleration only on Object. CARE also has the lowest evaluation cost, using 5,263 rather than 25,000 rollouts (78.9% fewer) and reducing H1 evaluation from 1,000 to 113 rollouts. Compared with the sequential baselines, it uses 610, 1,222, and 2,043 fewer rollouts than the plug-in e-process, sequential exact, and sequential HB, respectively. Thus, over the broader family, CARE provides both the widest certification coverage and the largest rollout saving.

4.4 Cross-Architecture and Cross-Domain Evaluation

CARE requires only paired terminal outcomes and measured inference cost; it does not depend on a particular policy architecture or acceleration mechanism. We evaluate this method on the flow-matching VLA policies π_0 and $\pi_{0.5}$, and on language-model agents with Qwen3.5-9B and Llama-3.1-8B in Crafter (Figure 3, Appendix C).

Flow-step selection for $\pi_{0.5}$. Beyond OpenVLA-OFT, we evaluate $\pi_{0.5}$ and use its default ten-step inference with action horizon H8 as the reference. The candidate family contains the same policy with 2, 4, or 6 flow-refinement steps, ordered from lowest to highest computation. For each LIBERO suite, we use 250 task-stratified episodes, examine the evidence every 50 episodes, and control the three-candidate family at $\alpha = 0.05$. Exhaustive evaluation would require 1,000 rollouts per suite.

Table 5 shows that CARE generalizes to a different VLA architecture. It selects the two-step policy on Spatial and Object, reducing flow computation by $5\times$, and the four-step policy on Goal, reducing it by $2.5\times$. On Long, CARE stops all three candidates early and retains the reference, requiring 464 rather than 1,000 rollouts. Across four suites, it uses 1,250 candidate rollouts and only 26 reference rollouts owing to failure-triggered evaluation, 68.1% fewer than exhaustive evaluation.

Compute-reduced language-model agents in Crafter. Next, we evaluate whether CARE extends beyond robotic policies. Qwen3.5-9B and Llama-3.1-8B agents act in Crafter using learned schedules that reduce language-model computation on the `collect_wood` and `collect_drink` tasks. Each schedule is trained before certification and evaluated on a reserved stream. As shown in Table 6, both models certify their `collect_wood` schedules on every seed while reducing compute by 25.1%–30.6%, and the `collect_drink` schedules certify on two of three seeds with smaller reductions.

Table 5: **Certified flow-step selection for $\pi_{0.5}$ on LIBERO.** CARE certifies reduced-step policies on Goal, Spatial, and Object and retains the ten-step reference on Long, using 1,276 rather than 4,000 rollouts (68.1% fewer). Compute and speedup are relative to the ten-step reference.

Suite	Selected policy	Compute (% ref.)	Speedup (×)	Total rollouts	Rollout saving (%)
Goal	Four-step	40	1.26	305	69.5
Spatial	Two-step	20	1.45	254	74.6
Object	Two-step	20	1.45	253	74.7
Long	Reference	100	1.00	464	53.6
			Total	**1,276**	**68.1**

Together, the $\pi_{0.5}$ and Crafter results directly answer **Q4**, confirming that CARE supports different policy architectures, compute controls, benchmarks, and task domains.

4.5 Design Analysis and Ablations

We analyze five components of CARE in two groups. The validity analysis evaluates (1) paired episode-level outcomes, (2) family correction when selecting among multiple candidates, and (3) task-balanced acquisition at sequential looks. The efficiency analysis isolates (4) sequential fastest-first candidate evaluation and (5) failure-triggered reference evaluation. Complete settings and results are in Appendix B.1.

Validity analysis. Local acceleration effects do not consistently predict whole-episode AIF. For example, the local and episode-level rates are 4.3% and 1.0% on Object, compared with 9.5% and 2.7% on Long (Table 7), which supports certifying complete paired episodes rather than individual decisions. Family correction is also necessary (Table 8): in a controlled ten-candidate simulation, uncorrected selection deploys an over-budget policy in 46.0% of trials, whereas family-corrected testing remains at or below 0.15%. Finally, balanced round-robin acquisition (Table 9) stays within the candidate-level error budget for every tested heterogeneous task-risk profile, whereas task-by-task acquisition can certify an over-budget policy before its high-risk tasks are observed.

Table 6: **Certified compute reduction for Crafter agents.** `collect_wood` schedules certify on all seeds at 69.4%–74.9% of reference compute; `collect_drink` schedules certify on two of three with compute reduced.

Model	Task	Certified seeds	Compute (% ref.)
Qwen3.5-9B	Wood	3/3	69.4
Qwen3.5-9B	Drink	2/3	85.8
Llama-3.1-8B	Wood	3/3	74.9
Llama-3.1-8B	Drink	2/3	98.6

Efficiency analysis. Sequential fastest-first evaluation reduces candidate rollouts from 8,000 to 1,950 by stopping candidates that can no longer be certified. Failure-triggered evaluation further reduces H1 rollouts from 1,000 to 47 by running the reference only after candidate failures. Together, the two components reduce total evaluation from 9,000 to 1,997 rollouts, a 77.8% reduction, and wall-clock time from 38.6 to 7.0 hours, an 81.9% reduction (Table 10).

5 Conclusions

We introduce CARE, an approach for certified accelerator selection based on acceleration-induced failure, a paired, episode-level risk that cannot in general be identified from average success or per-step measurements alone. CARE returns the fastest candidate whose risk can be certified below a user-specified budget and retains the reference when the available evidence is insufficient. Task-balanced sequential testing and failure-triggered reference evaluation substantially reduce the closed-loop rollout cost of certification while preserving the deployment guarantee. Across OpenVLA-OFT and $\pi_{0.5}$ on LIBERO and language-model agents in Crafter, CARE applies across different policy classes and acceleration mechanisms, recovers substantial acceleration when supported by the data, and avoids the over-budget selections produced by common heuristic deployment rules. These results suggest that acceleration need not be treated only as a speed–accuracy trade-off: it can instead be selected with explicit, finite-sample control over the failures it introduces.

REFERENCES

Anastasios Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *International conference on learning representations*, volume 2024, pp. 55198–55218, 2024.

Anastasios N Angelopoulos. Conformal risk control for non-monotonic losses. *arXiv preprint arXiv:2602.20151*, 2026.

Anastasios N Angelopoulos, Stephen Bates, Emmanuel J Candès, Michael I Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *The Annals of Applied Statistics*, 19(2):1641–1662, 2025.

Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

Vidmantas Bentkus. On hoeffding's inequalities. *The Annals of Probability*, 32(2):1650–1673, 2004.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

Steven R Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Metod Jazbec, Alexander Timans, Tin H Veljković, Kaspar Sakmann, Dan Zhang, Christian A Naesseth, and Eric Nalisnick. Fast yet safe: Early-exiting with risk control. *Advances in Neural Information Processing Systems*, 37:129825–129854, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Bracha Laufer-Goldshtein, Adam Fisch, Regina Barzilay, and Tommi Jaakkola. Efficiently controlling multiple risks with pareto testing. *arXiv preprint arXiv:2210.07913*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference. *arXiv preprint arXiv:2511.16449*, 2025.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 10832–10851, 2026.

Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023. doi: 10.1214/23-STS894.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 2380–2388. PMLR, 2015.

Leslie G Valiant. A theory of the learnable. In *Foundations of Computation and Machine Learning: The Work of Leslie Valiant*, pp. 69–94. 2026.

Jean Ville. Etude critique de la notion de collectif, 1939.

Vladimir Vovk and Ruodu Wang. E-values: Calibration, combination and applications. *The Annals of Statistics*, 49(3):1736–1754, 2021.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *Advances in Neural Information Processing Systems*, 38:164448–164473, 2026.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-congruent training: Towards regression-free model updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14299–14308, 2021.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

Tong Zheng, Chengsong Huang, Runpeng Dai, Yun He, Rui Liu, Xin Ni, Huiwen Bao, Kaishen Wang, Hongtu Zhu, Jiaxin Huang, et al. Parallel-probe: Towards efficient parallel thinking via 2d probing. *arXiv preprint arXiv:2602.03845*, 2026.

A THEORETICAL DETAILS

This appendix gives the full procedure (App. A.1), the assumptions and the p-value (App. A.2), the validity proof (App. A.3), the supermartingale property it relies on (App. A.4), the identification results (App. A.5), and the retention bounds (App. A.6).

A.1 THE SEQUENTIAL PROCEDURE

Algorithm 1 summarizes CARE. Candidates are evaluated in fastest-first order, each on the calibration units round by round. The reference is executed on a unit only when the candidate fails there and the reference outcome is not yet cached (lines 8–15). At each inspection point, the candidate is certified if its e-process reaches $1/\delta_j$ (lines 18–21), and abandoned if it can no longer reach this threshold even with no further losses (lines 22–25). The fixed-sample version is the special case with a single inspection at $n_K = n$, where the e-process test is replaced by $\widehat{p}_j \le \delta_j$.

Algorithm 1 CARE: certify-then-select with task-balanced rounds and failure-triggered reference evaluation

Require: Reference policy π_0; candidates $\Lambda = \{\lambda_1, \dots, \lambda_M\}$; risk budget α; error levels $\{\delta_j\}_{j=1}^{M}$ with $\sum_j \delta_j \le \delta$; grid $\mathcal{Q} \subset [0, \alpha)$; profiling set $\mathcal{D}_{\text{prof}}$; calibration units $U_1, \dots, U_{n_K}$ arranged in K rounds of m units per task, with inspection points $n_k = kB$, $B = mT$

Ensure: Selected configuration $\widehat{\lambda} \in \Lambda \cup \{\lambda_0\}$

1: Measure latencies on $\mathcal{D}_{\text{prof}}$ and index Λ from fastest (λ_1) to slowest (λ_M)
2: Initialize the reference-outcome cache $\mathcal{C} \leftarrow \emptyset$
3: **for** $j = 1, \dots, M$ **do**
4: $\quad f_j \leftarrow 0$ ▷ number of AIF losses of λ_j so far
5: $\quad$ **for** $k = 1, \dots, K$ **do**
6: $\quad\quad$ **for** $i = n_{k-1} + 1, \dots, n_k$ **do**
7: $\quad\quad\quad$ Run π_{λ_j} on U_i and observe $Y_j(U_i)$
8: $\quad\quad\quad$ **if** $Y_j(U_i) = 1$ **then**
9: $\quad\quad\quad\quad L_{j,i} \leftarrow 0$ ▷ candidate succeeds: loss is 0 whatever π_0 does
10: $\quad\quad\quad$ **else**
11: $\quad\quad\quad\quad$ **if** U_i is not in the reference cache $\mathcal{C}$ **then**
12: $\quad\quad\quad\quad\quad$ Run π_0 on U_i and store $\mathcal{C}[U_i] \leftarrow Y_0(U_i)$ ▷ failure-triggered
13: $\quad\quad\quad\quad$ **end if**
14: $\quad\quad\quad\quad L_{j,i} \leftarrow \mathcal{C}[U_i]$ ▷ candidate fails: use cached reference outcome
15: $\quad\quad\quad$ **end if**
16: $\quad\quad\quad f_j \leftarrow f_j + L_{j,i}$
17: $\quad\quad$ **end for**
18: $\quad\quad$ Compute $E_j(n_k)$ from f_j by Eq. 2
19: $\quad\quad$ **if** $E_j(n_k) \ge 1/\delta_j$ **then**
20: $\quad\quad\quad$ **return** $\widehat{\lambda} = \lambda_j$ ▷ certified; fastest certified candidate
21: $\quad\quad$ **end if**
22: $\quad\quad E_j^{\max} \leftarrow \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \left(\frac{q}{\alpha}\right)^{f_j} \left(\frac{1-q}{1-\alpha}\right)^{n_K - f_j}$ ▷ no further losses
23: $\quad\quad$ **if** $E_j^{\max} < 1/\delta_j$ **then**
24: $\quad\quad\quad$ **break** ▷ early exit: λ_j can no longer be certified
25: $\quad\quad$ **end if**
26: $\quad$ **end for**
27: **end for**
28: **return** $\widehat{\lambda} = \lambda_0$ ▷ no candidate certified: keep the reference

A.2 ASSUMPTIONS AND THE HOEFFDING–BENTKUS p-VALUE

We write $R_t(\lambda) = \mathbb{E}[L_\lambda(U) \mid \tau = t]$ for the per-task risk, so that $R(\lambda) = \frac{1}{T} \sum_t R_t(\lambda)$. The guarantees rest on three assumptions.

(A1) *Pre-specification.* The candidates, α, δ and its split $\{\delta_j\}$, the round schedule, the grid $\mathcal{Q}$, and the fastest-first order (measured on $\mathcal{D}_{\text{prof}}$) are fixed before any calibration outcome is observed.

(A2) *Sampling.* Calibration units are independent, and scenes are drawn i.i.d. within each task from the deployment distribution. Fixed-sample units are either i.i.d. from the equal-weight

task mixture or stratified with n/T units per task; sequential rounds contain exactly m units of every task.

(A3) *Executions.* Each execution launched from a unit depends only on that unit and on randomness independent of all other executions. Residual simulator nondeterminism is allowed and is part of the law of $(Y_0(U), Y_\lambda(U))$ that defines R.

The Hoeffding–Bentkus p-value. For independent $[0,1]$-valued $Z_1, \dots, Z_n$ with empirical mean $\widehat{Z}$ and a threshold $\theta \in (0,1)$,

$$\widehat{p}(\widehat{Z};\theta) = \min\Big\{1,\ e\ \Pr\big[\mathrm{Bin}(n,\theta) \le \lceil n\widehat{Z}\rceil\big],\ \exp\big(-n\,\mathrm{KL}(\widehat{Z}\wedge\theta\,\|\,\theta)\big)\Big\}, \tag{3}$$

where $a \wedge b = \min\{a,b\}$, $\mathrm{KL}(a\|b) = a\log\frac{a}{b} + (1-a)\log\frac{1-a}{1-b}$, and e is Euler's number. If the average mean $\frac{1}{n}\sum_i \mathbb{E}[Z_i]$ is at least θ, the binomial term (Bentkus's inequality (Bentkus, 2004)) and the exponential term (Hoeffding's inequality (Hoeffding, 1963)) both upper bound the distribution function $F(z) = \Pr(\widehat{Z} \le z)$ at every z; neither inequality requires identically distributed variables. Hence $\widehat{p} \ge F(\widehat{Z})$, and $\Pr(\widehat{p} \le s) \le s$ for all s (Bates et al., 2021). CARE uses $\widehat{p}_j = \widehat{p}(\widehat{R}_j;\alpha)$ with $Z_i = L_{j,i}$. Under (A2) the means of the $L_{j,i}$ average to $R(\lambda_j)$, both for i.i.d. and for stratified sampling, so $\widehat{p}_j$ is valid under H_j.

A.3 Validity of CARE

The proofs use one device. By (A3), all execution randomness can be drawn before the procedure starts, which fixes an *outcome table* with entries $Y_0(U_i)$ and $Y_j(U_i)$ for every calibration unit and every candidate. The procedure only decides which entries to reveal; revealing an entry later, or not at all, does not change it, and revealed entries have the same joint law as the outcomes of an actual run. We define $L_{j,i}$ and $E_j(n_k)$ from the full table, for every j and $k \le K$.

Lemma 1 (Failure-triggered reference evaluation)**.** *Under (A3), on the outcome table: (i) every loss computed with failure-triggered evaluation equals $L_{j,i} = Y_0(U_i)(1 - Y_j(U_i))$, so the evaluated units, all certification decisions, and $\widehat{\lambda}$ coincide with those of exhaustive paired evaluation; (ii) the reference is executed at most once per unit and only on units where some evaluated candidate fails, so $N_{\mathrm{ref}} = |\mathcal{C}| \le \min\{n_K, \sum_j F_j\}$, where F_j is the number of failures of λ_j on the units it is evaluated on.*

Proof. (i) If $Y_j(U_i) = 1$, the procedure assigns $L_{j,i} = 0$, which is correct for either value of $Y_0(U_i)$. If $Y_j(U_i) = 0$, the procedure uses the cached reference outcome $\mathcal{C}[U_i]$. If U_i has not yet been cached, π_0 is executed once on U_i and the result is stored as $\mathcal{C}[U_i] = Y_0(U_i)$. Hence, in either case, $L_{j,i} = \mathcal{C}[U_i] = Y_0(U_i) = Y_0(U_i)(1 - Y_j(U_i))$, so every computed loss is exactly the same as under exhaustive paired evaluation. Since the next unit evaluated and all subsequent stopping and certification decisions depend only on the losses observed so far, the failure-triggered procedure and exhaustive paired evaluation make the same decisions by induction.

(ii) The reference is executed on U_i only after a candidate fails on that unit and only if U_i is not already present in the reference cache $\mathcal{C}$. Once executed, its outcome is cached and reused for all later candidates. Therefore, each unit triggers at most one reference execution, and every reference execution corresponds to a distinct candidate failure. Thus, $N_{\mathrm{ref}} = |\mathcal{C}| \le \min\{n_K, \sum_j F_j\}$. □

Theorem 1 (Risk control)**.** *Under (A1)–(A3),* CARE*, in its fixed-sample and sequential versions and with fastest-first search, early exit, and failure-triggered reference evaluation, satisfies $\Pr\big(R(\widehat{\lambda}) \ge \alpha\big) \le \delta$.*

Proof. By Lemma 1(i), we may analyze exhaustive paired evaluation on the outcome table.

Each test has level δ_j. In the fixed-sample version, $\widehat{p}_j$ is valid (App. A.2), so $\Pr_{H_j}(\widehat{p}_j \le \delta_j) \le \delta_j$. In the sequential version, $(E_j(n_k))_{k=0}^K$ is a non-negative supermartingale under H_j with $E_j(0) = 1$ (Lemma 2), so Ville's inequality (Ville, 1939) gives $\Pr_{H_j}(\max_{k\le K} E_j(n_k) \ge 1/\delta_j) \le \delta_j$. A candidate is certified only at an inspection where $E_j(n_k) \ge 1/\delta_j$, so this bound holds whatever data-dependent inspection evaluation stops at.

Early exit. For $q \in \mathcal{Q}$, write $a_q = q/\alpha < 1 < b_q = (1-q)/(1-\alpha)$. The q-term of Eq. 2 after n units with f losses is $a_q^f b_q^{n-f}$, which decreases in f and increases in n. Since losses only accumulate, every later value of E_j is at most $E_j^{\max}$, the value with no further losses at n_K (Algorithm 1, line 22). If $E_j^{\max} < 1/\delta_j$, the candidate would never be certified, so abandoning it does not change which candidates are certified.

Selection. By the union bound, the probability that some unsafe candidate ($R(\lambda_j) \geq \alpha$) is certified is at most $\sum_j \delta_j \leq \delta$. Otherwise, every certified candidate, in particular the deployed one, has $R < \alpha$, and the fallback has $R(\lambda_0) = 0$. □

A.4 Supermartingale Property under Task-Balanced Rounds

Let $\mathcal{F}_k$ be generated by the outcome-table rows of rounds $1, \ldots, k$, and for $q \in \mathcal{Q}$ let $M_{j,i}(q) = (q/\alpha)^{L_{j,i}} \big((1-q)/(1-\alpha)\big)^{1-L_{j,i}}$, so that the q-term of Eq. 2 is $E_j(n_k, q) = \prod_{i \leq n_k} M_{j,i}(q)$.

Lemma 2. *Under (A2), (A3), and $H_j : R(\lambda_j) \geq \alpha$, $(E_j(n_k))_{k \geq 0}$ is a non-negative supermartingale with respect to $(\mathcal{F}_k)$.*

Proof. Fix q and round k. Its units are fresh draws, independent of $\mathcal{F}_{k-1}$ and of each other, and unit i of task $t(i)$ has conditional loss mean $\mu_i = R_{t(i)}(\lambda_j)$. Since each round contains m units of every task, $\frac{1}{B}\sum_i \mu_i = R(\lambda_j) \geq \alpha$. With $c = \frac{q-\alpha}{\alpha(1-\alpha)} < 0$, $\mathbb{E}[M_{j,i}(q) \mid \mathcal{F}_{k-1}] \leq 1 + c(\mu_i - \alpha)$, with equality for binary losses; this holds for any $[0,1]$-valued loss because $a^z b^{1-z} \leq za + (1-z)b$ for $z \in [0,1]$, and the right-handside is non-negative. By conditional independence and the inequality of arithmetic and geometric means,

$$\mathbb{E}\Big[\prod_{i \in \text{round } k} M_{j,i}(q) \;\Big|\; \mathcal{F}_{k-1} \Big] = \prod_{i \in \text{round } k} \big(1 + c(\mu_i - \alpha)\big) \leq \big(1 + c(R(\lambda_j) - \alpha)\big)^B \leq 1. \tag{4}$$

Hence each $E_j(n_k, q)$ is a non-negative supermartingale, and so is their average $E_j(n_k)$. □

Balanced rounds are what make the average conditional mean of each round equal $R(\lambda_j)$; a round drawn only from low-risk tasks would have an increment with expectation above one.

A.5 Identification: Why Paired, Complete Episodes Are Needed

Marginal success rates. Let $s_0 = \Pr(Y_0{=}1)$, $s_\lambda = \Pr(Y_\lambda{=}1)$, and $p_{ab} = \Pr(Y_0{=}a, Y_\lambda{=}b)$ under the task mixture, so $R(\lambda) = p_{10}$. Fixing the marginals leaves one free parameter, p_{11}, which can take any value in $[\max\{0, s_0 + s_\lambda - 1\}, \min\{s_0, s_\lambda\}]$. Hence $R(\lambda) = s_0 - p_{11}$ can be any value in $[\max\{0, s_0 - s_\lambda\}, \min\{s_0, 1 - s_\lambda\}]$. For example, with $s_0 = s_\lambda = 0.9$, any $R(\lambda) \in [0, 0.1]$ is possible. The success gap satisfies $s_0 - s_\lambda = R(\lambda) - p_{01}$: episodes the candidate repairs (p_{01}) cancel episodes it breaks.

Individual decisions. Consider an accelerator that replaces the reference action with an accelerated action at a set of decisions $\mathcal{T}_\lambda \subseteq \{0, \ldots, H-1\}$. Its *single-decision effect* at $t \in \mathcal{T}_\lambda$ is the AIF loss of the controller that accelerates decision t alone and follows the reference elsewhere.

Theorem 2 (Single-decision effects do not determine AIF risk)**.** *There exist two accelerators on the same task, using the same accelerated action, whose single-decision effects are all zero, yet whose AIF risks are* 1 *and* 0*.*

Proof. Let the state be a scalar error $x_t \geq 0$ with $x_0 = 0$, and let an episode of $H \geq 2$ decisions succeed if and only if $x_H \leq 1$. The reference action sets $x_{t+1} = \max\{x_t - 1, 0\}$, so the reference keeps $x_t \equiv 0$ and succeeds. The accelerated action sets $x_{t+1} = x_t + 1$. Accelerating a single decision raises the error to 1, after which reference actions return it to 0 (or leave $x_H = 1$ if $t = H-1$), so every single-decision effect is zero. Accelerating every decision gives $x_H = H \geq 2$ and fails, so its risk is 1. Accelerating every other decision keeps $x_t \leq 1$ throughout and succeeds, so its risk is 0. □

The fully accelerated controller reaches states (persistent error) that no single-decision intervention from the reference trajectory reaches, which is why AIF risk must be measured on complete episodes.

A.6 Retention Bounds and Retention Budgets

Let $s_0 := \Pr(Y_0 = 1)$ and assume $s_0 > 0$. Retention is $\mathrm{Ret}(\lambda) = 1-\rho$ with $\rho = R(\lambda)/\Pr(Y_0{=}1) \in [0, 1]$. Because it is a ratio of two unknown quantities, we bound it by inverting a family of tests rather than dividing estimates.

Theorem 3 (Retention bound)**.** *For $\beta \in (0, 1]$ let $Z_\beta(U) = \big(L_\lambda(U) - \beta Y_0(U) + \beta\big)/(1 + \beta)$, and let $\widehat{p}(\beta) = \widehat{p}\big(\widehat{Z}_\beta; \frac{\beta}{1+\beta}\big)$ as in Eq. 3. Let $\mathcal{S} = \{\beta \in (0, 1] : \widehat{p}(\beta) > \delta'\}$ (with $\sup \emptyset = 0$) and let $\bar{b} \geq \sup \mathcal{S}$. Then $\Pr[\mathrm{Ret}(\lambda) \geq 1 - \bar{b}] \geq 1 - \delta'$. With $\delta' = \delta/M$ for every candidate, the bounds hold simultaneously, and hence for $\widehat{\lambda}$, with probability at least $1 - \delta$.*

Proof. Since $L_\lambda \leq Y_0$, the pair (Y_0, L_λ) is $(0, 0)$, $(1, 0)$, or $(1, 1)$, giving $Z_\beta = \frac{\beta}{1+\beta}$, 0, or $\frac{1}{1+\beta}$, so $Z_\beta \in [0, 1]$. Taking expectations, $\mathbb{E}[Z_\beta] = \big(R(\lambda) - \beta \Pr(Y_0{=}1) + \beta\big)/(1 + \beta)$, which equals $\frac{\beta}{1+\beta}$ exactly when $\beta = \rho$. If $\rho = 0$, the claim holds since $\bar{b} \geq 0$. If $\rho > 0$, the test at $\beta = \rho$ has a true null, so $\Pr(\widehat{p}(\rho) \leq \delta') \leq \delta'$ (App. A.2). Hence, with probability at least $1 - \delta'$, $\rho \in \mathcal{S}$, so $\rho \leq \bar{b}$. The simultaneous statement follows from the union bound. □

Computing $\bar{b}$. $\widehat{p}(\beta)$ is not monotone in β, because the ceiling in the binomial term jumps as $n\widehat{Z}_\beta$ crosses integers, so a bisection can return an invalid bound. Instead, we exclude β values where either term of Eq. 3 is at most δ'. Since $\widehat{Z}_\beta$ is monotone in β, the integer crossings have closed form, and between them the binomial term is monotone; the exponential term is excluded on a grid with an explicit Lipschitz constant. Scanning down from $\beta = 1$, $\bar{b}$ is the upper end of the first cell that cannot be excluded, so $\bar{b} \geq \sup \mathcal{S}$.

Certifying a retention budget. For a retention budget $1 - \beta$, let $\theta_\beta = \beta/(1 + \beta)$ and test, for each candidate, $H_j^{\mathrm{Ret}} : \mathrm{Ret}(\lambda_j) \leq 1 - \beta$. By the proof of Theorem 3, H_j^{Ret} holds if and only if $\mathbb{E}[Z_\beta] \geq \theta_\beta$, a mean test on the $[0, 1]$-valued loss Z_β. The fixed-sample test uses $\widehat{p}(\widehat{Z}_\beta; \theta_\beta)$, which is valid by App. A.2. The sequential test uses Eq. 2 with α replaced by θ_β and $f_j(n_k)$ by $\sum_{i \leq n_k} Z_{\beta,j,i}$; Lemma 2 still applies (see its proof), and early exit remains valid since $Z_\beta \geq 0$. Theorem 1 then holds with Ret in place of R: with probability at least $1 - \delta$, $\mathrm{Ret}(\widehat{\lambda}) > 1 - \beta$, and the fallback has $\mathrm{Ret}(\lambda_0) = 1$. Since Z_β depends on Y_0 even when the candidate succeeds, this mode runs the reference once on every calibration unit instead of using failure-triggered evaluation.

Retention under failure-triggered evaluation. The bound above needs Y_0 on every unit, which failure-triggered evaluation does not provide. In that mode we instead run the reference on n_0 calibration units designated in advance, whose empirical success rate $\widehat{s}_0$ gives, by Hoeffding's inequality, $\Pr(Y_0{=}1) \geq \ell_0 = \widehat{s}_0 - \sqrt{\ln(1/\delta_0)/(2n_0)}$ with probability at least $1 - \delta_0$. Combined with $R(\widehat{\lambda}) < \alpha$ (Theorem 1), the union bound gives $\mathrm{Ret}(\widehat{\lambda}) > 1 - \alpha/\ell_0$ with probability at least $1 - \delta - \delta_0$ whenever $\ell_0 > 0$ (and $\mathrm{Ret} = 1$ if $\widehat{\lambda} = \lambda_0$). These reference outcomes are cached and reused during certification. We use $\delta_0 = 0.01$ and split the remaining 0.04 across the thirty-two suite–candidate hypotheses, so the combined statement holds with probability at least 0.95.

B Additional Experimental Results

B.1 Design-Choice Validation and Cost Ablations

This section provides the complete analyses underlying Section 4.5. We first examine the three choices required for valid certification and then isolate the two mechanisms that reduce evaluation cost.

Episode-level outcomes. We first compare accelerating a single policy decision with applying acceleration throughout the complete episode. The former measures a local intervention, whereas the latter measures the deployment risk targeted by CARE. We show the results in Table 7.

Table 7: **Local acceleration effects do not determine whole-episode AIF.** The relationship varies across suites because an accelerated action can alter subsequent states, observations, and actions.

Suite	One accelerated decision	Whole-episode AIF
Goal	2.3%	2.6%
Spatial	0.7%	0.8%
Object	4.3%	1.0%
Long	9.5%	2.7%

The two quantities are close on Goal and Spatial but differ by factors of 4.31 and 3.52 on Object and Long. Their relationship is therefore not stable across deployments. This result supports measuring the terminal outcomes of complete paired episodes rather than using local confidence or disagreement signals as a proxy for AIF.

Family correction. We next isolate the effect of searching over multiple candidate policies. We simulate ten candidates with known risks around $\alpha = 0.1$, use 200 calibration episodes per trial, and repeat the experiment for 4,000 trials. Each method selects from the same candidate family and observations. We present the results in Table 8.

Table 8: **Family correction prevents selection-induced risk inflation.** Searching over ten candidates without correction deploys an over-budget policy in 46.0% of trials, whereas family-corrected testing remains at or below 0.15%.

Selection rule	Over-budget deployment
Uncorrected empirical selection	46.0%
Family-corrected testing	$\leq 0.15\%$

The uncorrected rule repeatedly selects candidates whose empirical risks appear favorable because of sampling variation. Correcting the complete family prevents this selection effect from invalidating the deployment guarantee.

Task-balanced acquisition. The deployment risk averages over a fixed mixture of tasks. Sequential certification must therefore preserve that mixture at every interim look. We compare the balanced round-robin schedule used by CARE with task-by-task acquisition, which completes one task before proceeding to the next. Each entry in Table 9 is the exact probability of certifying an over-budget candidate.

Table 9: **Balanced acquisition preserves sequential validity.** Balanced rounds remain below the candidate-level error budget for every tested task-risk profile, whereas task-by-task ordering can certify an over-budget policy before its high-risk tasks are evaluated.

Task-risk profile	Balanced rounds	Task by task
Eight safe, two risky; $R = 0.050$	1.84×10^{-4}	1.00
Graded; $R = 0.050$	4.37×10^{-4}	1.42×10^{-2}
Homogeneous; $R = 0.050$	4.77×10^{-4}	4.77×10^{-4}
One high-risk task; $R = 0.060$	1.08×10^{-6}	1.00
Six safe, four risky; $R = 0.052$	2.43×10^{-4}	1.00

Balanced acquisition remains valid even when AIF risk varies sharply across tasks. Task-by-task ordering can initially expose only low-risk tasks and certify a policy before observing the tasks responsible for its true deployment risk.

Evaluation-cost components. Finally, we separate the two mechanisms used by CARE to reduce certification cost. We present the results in Table 10. Sequential fastest-first evaluation tests policies in increasing order of measured inference cost and stops a candidate when certification is no longer

attainable. Failure-triggered evaluation runs H1 only when a candidate fails, because the paired AIF outcome is already zero whenever the candidate succeeds.

Table 10: **Component-wise ablation of certification cost.** Sequential fastest-first evaluation reduces candidate rollouts, while failure-triggered evaluation reduces H1 rollouts. Combining both components reduces total rollouts by 77.8% and wall-clock time by 81.9% relative to fixed-n test-all evaluation.

Procedure	Candidate rollouts	H1 rollouts	Total rollouts	Wall-clock time
Fixed-n test-all	8,000	1,000	9,000	38.6 h
Fixed-n + failure-triggered H1	8,000	349	8,349	33.1 h
Sequential fastest-first + full H1	1,950	1,000	2,950	16.3 h
CARE	**1,950**	**47**	**1,997**	**7.0** h

Table 11: **Sensitivity to risk budget and calibration size.** A larger risk budget or more calibration evidence reduces fallback and increases speedup, while CARE maintains an observed over-budget rate of at most 0.02%.

$\alpha(\%)$	n	Over budget (%)	Fallback (%)	Speedup ($\times$)
2.0	250	0.00	100.0	1.00
2.5	250	0.00	100.0	1.00
3.0	250	0.00	69.5	3.51
5.0	250	0.00	11.7	8.88
10.0	250	0.00	0.0	11.25
2.5	500	0.02	56.3	4.59
2.5	1,000	0.01	15.0	8.48
2.5	2,000	0.00	1.4	9.99

Sequential fastest-first evaluation provides the largest individual saving, reducing candidate rollouts by 75.6%. Failure-triggered evaluation provides a complementary saving by reducing reference evaluation to 47 rollouts after sequential candidate stopping. Together, the two components produce the full 77.8% rollout reduction achieved by CARE.

B.2 Sensitivity Analysis

We examine how the risk budget α and calibration size n affect CARE for sensitivity analysis. We report over-budget deployment, fallback to H1, and mean speedup relative to H1.

Table 11 shows the results. At the default setting ($\alpha = 5.0\%$, $n = 250$), CARE achieves an $8.88\times$ mean speedup with 11.7% fallback and no observed over-budget deployment. Tightening α increases fallback, whereas relaxing it allows faster policies to be certified. Under the stricter $\alpha = 0.025$ budget, increasing n from 500 to 2,000 reduces fallback from 56.3% to 1.4% and raises mean speedup from $4.59\times$ to $9.99\times$. Thus, CARE becomes less conservative as the budget or calibration evidence grows, while maintaining risk control. The default setting provides a practical operating point considering evaluation cost, while CARE adapts robustly to other risk budgets and calibration sizes.

C Benchmarks

LIBERO. LIBERO (Liu et al., 2023) is a simulated tabletop manipulation benchmark in which a Franka Panda arm follows language instructions. We use its four ten-task suites (Figure 2): *Spatial* varies object layouts, *Object* varies the target object, *Goal* varies the goal in a fixed scene, and *Long* chains several subgoals. Each task provides 50 fixed initial states, giving 500 scenes per suite, and an episode succeeds if the task's goal conditions are met before the step limit.

Crafter. Crafter (Hafner, 2021) is a procedurally generated 2D survival game with discrete actions and a status bar tracking health, food, drink, energy, and inventory (Figure 3). We use two

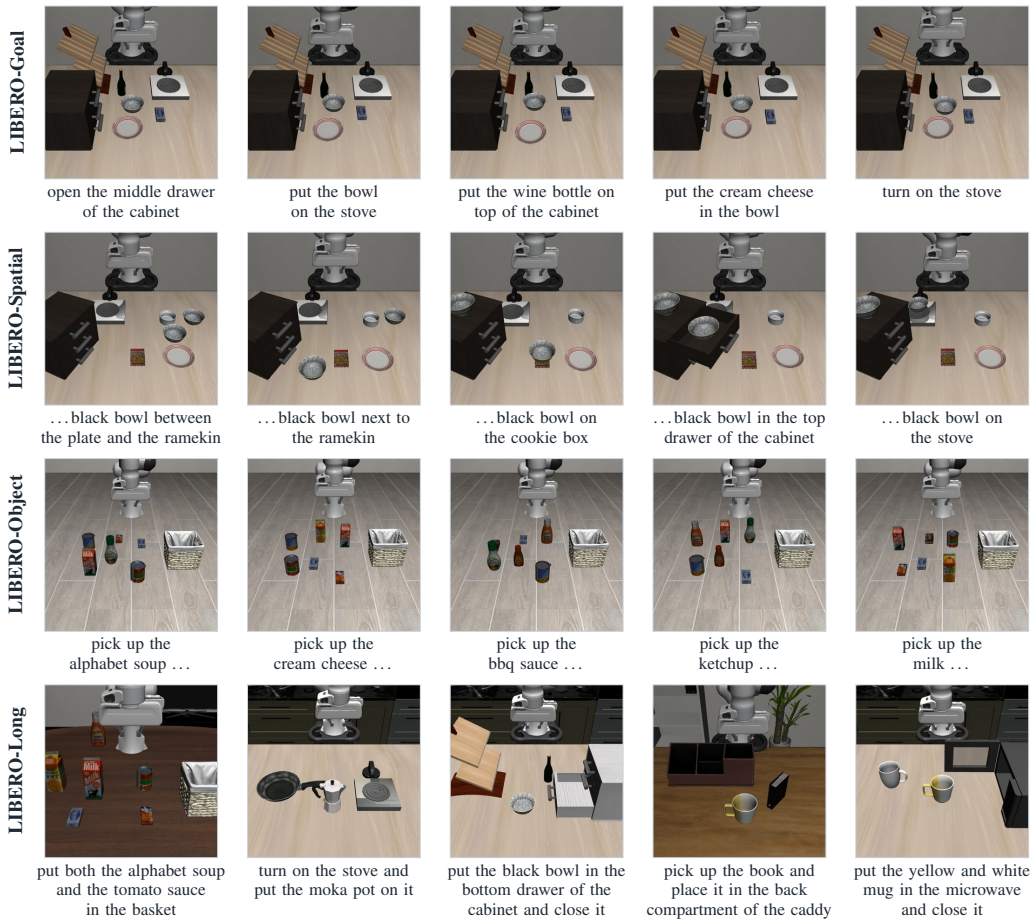

Figure 2: **LIBERO task suites.** Initial scenes of five of the ten tasks in each suite, rendered from the third-person agent-view camera, with the language instruction below each scene. Spatial instructions have the form "pick up the black bowl ... and place it on the plate" and Object instructions "pick up the ... and place it in the basket"; only the distinguishing part is shown. Each evaluation unit fixes a task, one initial scene, and the random seeds.

achievements as tasks, `collect_wood` and `collect_drink`, and an episode succeeds if the agent unlocks the achievement within the step budget. It tests whether CARE applies unchanged to language-model agents.

D AI ASSISTANCE

We used AI assistance to aid and polish writing, including Abstract, Section 1 Introduction and 2 Related Work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of AI.

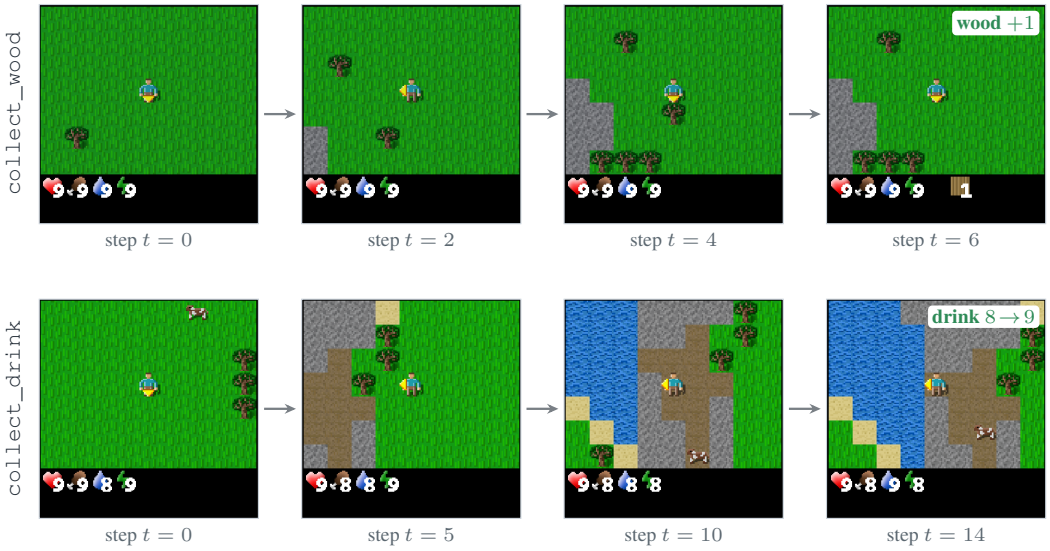

Figure 3: **Crafter tasks.** Example episodes of `collect_wood` (top) and `collect_drink` (bottom). The status bar shows health, food, drink, and energy, followed by the inventory; an episode succeeds once wood enters the inventory or the agent drinks water. Frames are rendered from the environment with a scripted policy for illustration and are not rollouts of the evaluated language-model agents.