Inference-Time Scaling Laws for Embodied Agents: Power Laws, Optimality, and Adaptive Deliberation under Latency Constraints

Liz Lemma Future Detective

January 17, 2026

Abstract

Neural scaling law work in robotics has focused almost entirely on training-time resources (data, parameters, FLOPs) and reports little about inference-time compute. Yet 2026-era robot systems increasingly rely on deliberation at test time: sampling multiple action plans, verifying safety/feasibility, running short world-model rollouts, or self-consistency over language-conditioned policies—often under strict latency and energy budgets. We introduce a clean candidate-and-verify formalization of test-time compute for embodied agents and prove that expected downstream error can exhibit power-law scaling in inference compute. Our key observation is that power laws arise generically when episode difficulty induces a heavy tail of small per-sample success probabilities. Under mild assumptions on the density of the latent success probability near zero, we show $\mathbb{E}[\mathrm{Err}(k)] = \Theta(k^{-a})$ with matching upper and lower bounds, making the exponent information-theoretically optimal. We then study adaptive allocation of a global compute budget across episodes and give approximation/online guarantees for compute controllers that concentrate deliberation on hard instances. Finally, we outline an experimental protocol wrapping an open-source VLA policy with a deliberation layer (plan sampling + verification and optional fast world-model simulation) to measure exponents, identify the regime where test-time compute outperforms parameter scaling, and connect these findings to robotics scaling-law meta-analyses that currently lack compute and inference axes.

Table of Contents

ing/compute reporting; contrast with meta-analysis findings emphasizing missing compute studies.

3. 3. Problem Setup and Computational Model: candidate generation, verification, and compute budget; formal definition of k-sample deliberation; deployment-level compute allocation across episodes.

4. 4. From Episode Difficulty to Power-Law Inference Scaling: latent success probability model; heavy-tail condition; derivation of $\Theta(k^{-a})$ error scaling; interpretation of exponent as difficulty mass near zero.

5. 5. Tight Bounds and Minimax Optimality: matching lower bounds for any k-limited algorithm; robustness to verification noise and dependence; extensions to imperfect critics/world models.

6. 6. Phase Transition Between Model Scaling and Inference Scaling: formal marginal-return comparison; conditions for existence of N^*; guidance for resource allocation (train vs infer) under latency constraints.

7. 7. Adaptive Compute Allocation Algorithms: offline (budgeted) and online (bandit/optimal stopping) compute controllers; approximation guarantees; regret bounds; practical estimators for per-episode difficulty.

8. 8. Experimental Protocol (Optional but Strengthening): how to wrap an open-source VLA with sampling + critic + optional world-model rollouts; controlled sweeps over C_{test}; measuring exponents; identifying N^*; ablations for confounds.

9. 9. Discussion and Limitations: when power laws may break (saturation, correlated samples, nonstationary tasks); implications for safety and benchmarking; recommendations for reporting inference compute.

10. 10. Conclusion: takeaways and open problems (joint train+test scaling surfaces for robotics; standardized inference compute accounting).

1 Introduction

Contemporary embodied agents increasingly couple a learned policy with a nontrivial amount of test-time computation: they sample candidate plans, score them with critics or verifiers, optionally simulate outcomes, and execute the best verified candidate. In manipulation and navigation this pattern appears under many names—sampling-based planning around a learned proposal distribution, LLM- or VLM-generated skill sequences with execution-time filtering, motion-planning with learned heuristics, and "plan–critic–act" wrappers. The commonality is that deployment performance is not determined by the parameters of a single forward pass alone, but by an *inference-time compute budget* that governs how many candidates are generated and checked before acting. We therefore treat test-time compute as a first-class scaling axis in robotics, on par with model size and training compute.

A training-only scaling viewpoint is incomplete for at least three reasons. First, robotics deployments are latency-limited: an agent must decide within a per-episode cap K (or a per-decision cap in receding-horizon settings), which makes the marginal value of additional deliberation itself a central quantity. Second, deployments are often budget-limited at the system level: fleets of robots share compute, power, or server capacity, inducing a global constraint B over a window of T episodes. Third, the empirical distribution of task difficulty is heterogeneous. For many episodes a competent base model π_N produces a correct plan quickly, while a minority of "hard" episodes require repeated resampling, backtracking, or verification to succeed. This heterogeneity suggests that the appropriate performance metric is not merely the best achievable success at a fixed average compute, but the entire *compute–reliability curve* and the allocation policy that traces it.

We formalize these considerations through a candidate-and-verify abstraction. Fix a trained base model π_N (parameters fixed during deployment). For each episode τ drawn from an environment distribution $\mathcal{T}$, the agent can spend k units of test-time compute to sample and verify k candidate plans $y \sim S_N(\cdot \mid \tau)$, where S_N is the candidate generator induced by π_N. A verifier $V(y, \tau)$ accepts feasible candidates; under sound verification, the episode succeeds if at least one accepted candidate is found and executed. Conditional on τ, this reduces the episode to a latent success probability $P_{\tau,N}$: a single candidate drawn from S_N passes verification and succeeds with probability $P_{\tau,N}$, and repeated candidate generations are (stylized as) conditionally i.i.d. given $P_{\tau,N}$. Under this model, spending k compute yields success probability $1 - (1 - P_{\tau,N})^k$, so compute yields diminishing returns even before introducing any additional structure.

The central question then becomes: *how does expected error scale with k under realistic heterogeneity in $P_{\tau,N}$?* A key empirical regularity across domains is that error is often governed by a heavy tail of hard instances. We capture this by assuming that, across episodes, the density of $P_{\tau,N}$ near 0

behaves as

$$f_N(p) \;=\; c_N p^{a_N - 1}(1 + o(1)) \qquad (p \downarrow 0),$$

for constants $a_N > 0$ and $c_N > 0$ depending on model size N. Intuitively, small p corresponds to episodes for which the proposal distribution seldom generates a viable plan; these episodes dominate the error for large k. Under this assumption, a standard Laplace-type calculation shows that the expected error after k candidate-and-verify steps obeys a power law in test-time compute,

$$\mathbb{E}[\mathrm{Err}_N(k)] \;=\; \mathbb{E}[(1 - P_{\tau,N})^k] \;=\; \Theta(k^{-a_N}),$$

with constants determined by (a_N, c_N). Thus, in this regime, improvements from additional test-time computation are predictable and quantifiable: doubling k multiplies error by approximately 2^{-a_N} at sufficiently large k. We emphasize that the exponent a_N is a property of the *instance distribution* and the *proposal mechanism* induced by π_N, not merely of the verifier or the planner.

A second question is whether smarter test-time algorithms can beat this exponent. In the absence of additional structure beyond i.i.d. sampling and sound verification, the answer is negative: we show a minimax lower bound demonstrating that no algorithm restricted to k candidate generations can improve the k^{-a_N} exponent uniformly over all distributions with the stated tail behavior. This clarifies what is and is not possible by "better search" alone: improvements in constants may be available, but the heavy-tail exponent is information-theoretically pinned down unless one leverages extra signals (e.g., informative verifier scores, structured neighborhoods of candidates, or amortized warm-starting across episodes).

The candidate-and-verify viewpoint also clarifies compute allocation under shared budgets. Since per-episode success is concave in k (diminishing marginal gain $\Delta_N(k)$), allocating B compute units across T episodes becomes a monotone submodular maximization problem. When the latent $P_{\tau,N}$ values are known (offline), a greedy allocation that assigns each additional unit of compute to the episode with the largest current marginal gain achieves the classical $(1 - 1/e)$ approximation guarantee. In the realistic online setting, where $P_{\tau,N}$ is unknown and only partially revealed through accept/reject outcomes (or scores), this perspective motivates controllers that spend additional compute on episodes estimated to be in the low-P tail, subject to the latency cap K and the remaining global budget.

Contributions. Our contributions are therefore conceptual and analytic. We (i) isolate inference-time compute as an explicit resource in robotics deployments and propose a clean candidate-and-verify abstraction parameterized by (N, k) and the latent per-episode pass probability $P_{\tau,N}$; (ii) prove

a power-law scaling of expected error in test-time compute under a heavy-tail assumption on the distribution of $P_{\tau,N}$ near 0, and establish a matching minimax lower bound on the scaling exponent; (iii) connect global-budget compute allocation across episodes to submodular maximization, yielding approximation guarantees for greedy allocation under diminishing returns; and (iv) formalize a resource tradeoff between increasing model size N and increasing test-time compute k by comparing marginal returns, yielding a phase-transition criterion under standard assumptions on how (a_N, c_N) vary with N.

We view these results as a step toward making robotics evaluation protocols more compute-aware: reporting only a single success rate without specifying test-time search budget obscures a crucial axis of capability. In the next section we situate our formulation relative to prior work on scaling laws, planning wrappers for embodied agents, and verification-based control.

2 Related Work

Training-time scaling laws. A large body of work studies how predictive error or downstream performance varies with training compute, dataset size, and model size under fixed evaluation procedures. Classical empirical scaling laws for language models relate cross-entropy loss to power laws in model and data size, and identify compute-optimal training frontiers **??**. Related analyses extend to vision and multimodal models, and to transfer performance on downstream tasks, typically assuming that test-time computation is dominated by a single forward pass at fixed context length **?**. Our emphasis differs in that embodied deployment frequently introduces an additional, controllable inference-time loop (sampling, scoring, and optional simulation) whose computational cost is comparable to, or larger than, a forward pass, and whose effect on reliability depends on instance heterogeneity rather than solely on average-case error.

Test-time compute and inference-time scaling. In language-model inference, there is an explicit notion of allocating additional test-time compute by sampling multiple completions and selecting among them, as captured by metrics such as pass@k in code generation and by best-of-n selection more broadly **?**. More recent work investigates deliberate inference procedures that use additional tokens, additional samples, or structured search (e.g., self-consistency, reranking, and tree-based deliberation) to improve accuracy **??**. Although these methods are usually motivated in terms of reasoning or search, from a resource perspective they instantiate a common pattern: repeated proposal and selection under a compute cap. The present paper isolates the corresponding scaling axis in a setting where the proposals are plans or control sequences rather than text continuations, and where a verifier em-

bodies feasibility, safety, or goal satisfaction. We also stress that the relevant limiting factor in robotics is often latency (per-episode or per-decision), which changes the operational meaning of a compute–accuracy curve: the relevant question is not merely whether additional compute helps, but how error decays as a function of the allowed number of proposal–verification iterations.

Embodied agents with planning wrappers. Many embodied systems couple a learned policy with explicit planning, resampling, or receding-horizon control. Classical sampling-based motion planning and trajectory optimization methods (e.g., RRT variants, CEM-style optimization, MPPI) already fit a "generate candidates, score/feasibility-check, execute best" template, with learned components used as proposal distributions, cost models, or warm starts **??**. In learning-based robotics, this pattern appears in model predictive control with learned dynamics, in learned planners that generate candidate action sequences, and in hierarchical agents that plan in a discrete skill space and then execute low-level controllers **??**. More recently, large language and vision–language models are wrapped with task-and-motion planning, tool-use controllers, or iterative refinement loops that propose candidate skill programs and filter them through simulation, constraint checking, or execution feedback **??**. Across these lines, the computational burden at deployment is frequently dominated by repeated candidate generation and evaluation, but is seldom treated as an explicit scaling variable. Our abstraction aims to unify these wrappers at the level of resource accounting: each additional candidate consumes compute and yields diminishing marginal improvement, with the extent of improvement governed by the distribution of per-instance proposal quality.

Verification, critics, and safety filters. The role of a verifier in our model is related to several mechanisms used in robotics and reinforcement learning. In planning, feasibility checks include collision checking and constraint satisfaction, which can be exact (geometric) or approximate (learned) and are routinely used as hard filters. In RL and control, one sees learned value functions or Q-functions used for action selection, discriminator-based critics for goal satisfaction, and safety "shields" that reject unsafe actions based on reachability or barrier-function reasoning **??**. In language-grounded robotics, verifiers include affordance classifiers, predicate evaluators, and simulation-based checks that ensure that a proposed plan is executable in the current scene **?**. Our analysis is agnostic to whether V is learned or analytic, and instead makes explicit the operational assumption required for the candidate-and-verify reduction: that an accepted candidate succeeds with high probability (soundness, possibly up to a bounded false-positive rate). This viewpoint complements work that focuses on learning better critics by

clarifying when improvements should be attributed to changing the proposal distribution versus changing the verifier, and by highlighting that repeated proposal–verification can induce predictable compute–reliability curves even with fixed learned components.

Benchmarking practice and compute reporting. Robotics benchmarks typically report success rates, path efficiency, or reward under fixed evaluation pipelines, with limited standardization of test-time computation budgets. In contrast, several areas in machine learning explicitly report performance as a function of test-time sampling (e.g., pass@k) or inference-time budgets, and maintain leaderboards that separate model capacity from sampling-based ensembling. For embodied agents, comparisons are complicated by heterogeneous hardware, simulator speed, and differing planner/verifier implementations; nevertheless, the absence of explicit compute reporting makes it difficult to interpret gains from planning wrappers versus gains from better base models. Meta-analyses and survey-style syntheses of embodied evaluation protocols repeatedly note that compute and latency constraints are under-specified, and that claims of "better reasoning" or "better planning" are often confounded by unreported deliberation budgets and selection mechanisms. Our contribution is not a new benchmark, but a formal account of what compute reporting should enable: (i) a compute–reliability curve indexed by k under a fixed base model π_N, and (ii) principled allocation rules when compute is shared across episodes. This framing motivates the computational model introduced next, where we treat candidate generation and verification as explicit oracles and make the per-episode and global budget constraints part of the problem statement.

3 Problem Setup and Computational Model

We formalize deployment as a sequence of episodes (task instances) $\tau \sim \mathcal{T}$, each of which specifies an instruction, an environment configuration, and an initial state. A trained base model/policy π_N of fixed size N is held constant at deployment time. The agent may, however, expend additional *test-time compute* to improve reliability by iteratively proposing candidate plans and filtering them through a verifier before executing.

Candidate-and-verify interface. For each episode τ we assume access to two black-box procedures induced by π_N and the surrounding system: a *candidate generator* (sampler) and a *verifier*. The generator is a distribution $S_N(\cdot \mid \tau)$ over candidates y; a candidate may represent, depending on the application, a low-level action sequence, a high-level skill program, a receding-horizon control prompt, or any structured plan object that can be executed by the robot. A single generator call returns an i.i.d. sample

$y \sim S_N(\cdot \mid \tau)$. The verifier is a predicate or score function $V(y, \tau)$ that returns either (i) `accept/reject`, or (ii) a real-valued score that can be thresholded into accept/reject. The semantics of verification are that acceptance certifies feasibility/safety/goal-satisfaction for the current τ up to a known bounded error, which we absorb into the effective success probability defined below.

Compute accounting and per-episode cap. We measure test-time compute in units of one "generate+verify" attempt. Thus, spending $k \in \{0, 1, \ldots, K\}$ compute on an episode means performing at most k sequential iterations of sampling and verification, with a hard per-episode latency cap K. In settings where generation and verification costs differ materially (or where each candidate is additionally rolled out in a world model), we allow weighted costs; one can interpret a unit budget as w_g units per generation plus w_v per verification (plus optional simulation cost w_s), and absorb these into a normalized budget without changing the structure of the analysis. The only requirement for our results is that the algorithm be feasible: it may stop early but it cannot exceed K for any single episode.

k-sample deliberation as a sequential decision rule. A *k-sample deliberation* procedure for episode τ is any (possibly randomized) sequential rule that, at each iteration i, draws $y_i \sim S_N(\cdot \mid \tau)$, queries $V(y_i, \tau)$, and then either stops and executes some selected candidate (typically the first accepted one, or the best among accepted), or continues to the next iteration, until $i = k$ or the procedure stops early. If no acceptable candidate is found within the allotted budget, the agent executes a designated fallback behavior (e.g., the base policy without deliberation, a safe default, or a "no-op"), yielding success or failure depending on the domain; for the scaling analysis it is convenient to treat fallback as failure, since any nontrivial fallback can be incorporated into the sampler as an additional candidate with its own acceptance probability.

We write $\mathrm{Succ}_{\tau,N}(k) \in \{0, 1\}$ for the success indicator under model size N and deliberation budget k, and $\mathrm{Err}_{\tau,N}(k) = 1 - \mathrm{Succ}_{\tau,N}(k)$ for the corresponding error. The primary per-episode quantity of interest is the success probability $\mathbb{P}(\mathrm{Succ}_{\tau,N}(k) = 1)$ induced by the deliberation rule.

Latent per-sample success probability. The abstraction becomes especially transparent under the following stylized reduction. For a fixed (τ, N), define a latent parameter $P_{\tau,N} \in [0, 1]$ to be the probability that a single draw $y \sim S_N(\cdot \mid \tau)$ will be accepted by the verifier and succeed when executed. Importantly, $P_{\tau,N}$ is *episode-specific*: it captures instance difficulty and proposal quality, and it may be arbitrarily small on hard episodes. Under conditional independence of candidate draws given (τ, N), a canonical

deliberation rule is "sample until acceptance or budget exhaustion," in which case the probability of failure after k attempts is

$$\mathbb{P}(\mathrm{Err}_{\tau,N}(k) = 1 \mid P_{\tau,N}) = (1-P_{\tau,N})^k, \qquad \mathbb{P}(\mathrm{Succ}_{\tau,N}(k) = 1 \mid P_{\tau,N}) = 1-(1-P_{\tau,N})^k. \tag{1}$$

We emphasize that (1) should be read as a computational model rather than a literal claim about all systems: it asserts that, after all modeling choices (including any bounded verifier false positives/negatives) are absorbed into the effective event "a sampled candidate works," deliberation acts as repeated independent trials. This reduction is the basis for the compute–reliability curves studied in the sequel.

Deployment objective and global compute budgets. Over a deployment window of T episodes, we may additionally impose a global budget B on the total number of generate+verify iterations:

$$\sum_{t=1}^{T} k_{\tau_t} \le B, \qquad 0 \le k_{\tau_t} \le K \quad \text{for all } t. \tag{2}$$

The controller must choose k_{τ_t} online (before or during episode t) and then run a feasible deliberation procedure subject to that cap. The natural objective is to maximize expected total successes,

$$\max \sum_{t=1}^{T} \mathbb{E}[\mathrm{Succ}_{\tau_t,N}(k_{\tau_t})]\,, \tag{3}$$

or, equivalently, minimize expected total errors. When it is useful to treat compute as a soft cost rather than a hard constraint, we consider the Lagrangian form $\sum_t \mathbb{E}[\mathrm{Succ}_{\tau_t,N}(k_{\tau_t})] - \lambda \sum_t k_{\tau_t}$ for some $\lambda > 0$.

Within-episode stopping and across-episode allocation. Two levels of adaptivity are permitted. First, within a single episode we may stop early when an acceptable candidate is found, and more generally we may use verifier outputs (scores) to form a running estimate of difficulty (a proxy for $P_{\tau,N}$) and decide whether further sampling is worthwhile. Second, under the global constraint (2), we may allocate different caps k_τ to different episodes based on cheap pre-deliberation signals (e.g., perception uncertainty, language ambiguity, or a preliminary verifier probe). The analyses that follow treat these controllers as operating in a constrained sequential decision problem with limited feedback: the latent $P_{\tau,N}$ is unobserved, but its effect is partially revealed through accept/reject outcomes.

The remainder of the paper studies the consequences of (1) when $P_{\tau,N}$ varies substantially across τ. In particular, we will relate the decay of $\mathbb{E}[\mathrm{Err}_N(k)] = \mathbb{E}_{\tau\sim\mathcal{T}}[(1-P_{\tau,N})^k]$ to distributional mass near $P_{\tau,N} = 0$, and we will use the diminishing-returns structure implicit in (1) to justify principled compute allocation under (2).

4 From Episode Difficulty to Power-Law Inference Scaling

The per-episode model (1) isolates a single latent parameter $P_{\tau,N}$ that measures how frequently the candidate generator, coupled with verification, produces an executable solution on episode τ. To pass from a within-episode statement to a deployment-wide compute–reliability law, we must understand how $P_{\tau,N}$ varies across $\tau \sim \mathcal{T}$. In particular, the scaling with k is governed not by typical episodes but by the "hard" tail where $P_{\tau,N}$ is very small: these are precisely the instances on which many independent trials are required before an acceptable candidate appears.

Formally, for fixed model size N and deliberation budget k, the expected test-time error rate is

$$\mathbb{E}[\mathrm{Err}_N(k)] \;=\; \mathbb{E}_{\tau\sim\mathcal{T}}\Big[(1-P_{\tau,N})^k\Big]. \tag{4}$$

Writing P for a generic draw from the marginal distribution of $P_{\tau,N}$ over τ, with density f_N on $[0,1]$, we may express (4) as the integral

$$\mathbb{E}[\mathrm{Err}_N(k)] \;=\; \int_0^1 (1-p)^k f_N(p)\,dp. \tag{5}$$

The behavior of (5) as $k \to \infty$ depends on how much probability mass f_N assigns to small p. If P were bounded away from 0 almost surely, then $(1-P)^k$ would decay exponentially and so would $\mathbb{E}[\mathrm{Err}_N(k)]$. Conversely, if $\mathcal{T}$ includes a non-negligible fraction of episodes for which $P_{\tau,N}$ is arbitrarily close to 0, then those episodes dominate (5) and the decay becomes polynomial.

Our central regularity assumption is that the density near 0 behaves as a power law:

$$f_N(p) = c_N p^{a_N-1}(1+o(1)) \qquad \text{as } p \downarrow 0, \tag{6}$$

for constants $a_N > 0$ and $c_N > 0$. Although $P \in [0,1]$ is bounded, we adopt the standard heavy-tail terminology to emphasize that (6) allows substantial mass near 0. The parameter a_N is the effective "difficulty exponent": smaller a_N means more probability mass concentrated near $P = 0$, i.e., more episodes on which the generator almost never proposes a viable plan.

Under (6), the expected error exhibits a power-law decay in k. The derivation is a Laplace-type asymptotic controlled by the change of variables $u = kp$ and the elementary approximation $(1-p)^k \approx e^{-kp}$ for the values of p that matter when k is large. Indeed, for any fixed $\delta \in (0,1)$, we decompose (5) as

$$\int_0^1 (1-p)^k f_N(p)\,dp = \int_0^\delta (1-p)^k f_N(p)\,dp + \int_\delta^1 (1-p)^k f_N(p)\,dp. \tag{7}$$

The second term in (7) is negligible because $(1-p)^k \le (1-\delta)^k$ for $p \in [\delta, 1]$, hence

$$\int_\delta^1 (1-p)^k f_N(p)\, dp \le (1-\delta)^k, \tag{8}$$

which decays exponentially in k. The first term captures the contribution of the hard episodes. Using (6) and bounding $(1-p)^k$ between $e^{-kp/(1-p)}$ and e^{-kp} for $p \in (0, \delta)$ (or simply using $(1-p)^k = e^{k\log(1-p)}$ and $\log(1-p) = -p + O(p^2)$), we obtain

$$\int_0^\delta (1-p)^k f_N(p)\, dp = (1+o(1)) \int_0^\delta e^{-kp}\, c_N p^{a_N - 1}\, dp. \tag{9}$$

The remaining integral is explicit after the substitution $u = kp$:

$$\int_0^\delta e^{-kp} c_N p^{a_N-1}\, dp = c_N k^{-a_N} \int_0^{k\delta} e^{-u} u^{a_N-1}\, du = c_N k^{-a_N} \Gamma(a_N)\,(1+o(1)), \tag{10}$$

since $\int_0^{k\delta} e^{-u} u^{a_N-1} du \to \Gamma(a_N)$ as $k \to \infty$. Combining (8)–(10) yields the asymptotic law

$$\mathbb{E}[\mathrm{Err}_N(k)] = c_N \Gamma(a_N) k^{-a_N}\,(1+o(1)). \tag{11}$$

Thus the expected error is $\Theta(k^{-a_N})$, and, equivalently, the expected success satisfies

$$\mathbb{E}[\mathrm{Succ}_N(k)] = 1 - c_N \Gamma(a_N) k^{-a_N}\,(1+o(1)). \tag{12}$$

This conclusion is insensitive to the behavior of f_N away from 0; the entire asymptotic is determined by the small-p region, reflecting the fact that sufficiently easy episodes are solved quickly and cease to influence the marginal gains at large k.

We emphasize the operational interpretation of the exponent a_N. Solving (11) for the compute required to reach an average error level ϵ gives

$$k(\epsilon) \asymp \left(\frac{c_N \Gamma(a_N)}{\epsilon}\right)^{1/a_N}. \tag{13}$$

Hence when a_N is small (substantial mass of nearly-impossible episodes), reducing ϵ requires a rapidly increasing deliberation budget; when a_N is large (few extremely hard episodes), additional compute yields steep reliability gains. Empirically, (11) predicts that a log–log plot of $\mathbb{E}[\mathrm{Err}_N(k)]$ versus k is approximately linear for large k, with slope $-a_N$, providing a direct route to estimating the difficulty tail from deployment traces.

Finally, we remark on the boundary cases suggested by (6). If $f_N(p)$ vanishes near 0 fast enough (e.g., $P_{\tau,N} \ge p_0 > 0$ almost surely), then the decay is exponential rather than polynomial. If there is an atom at $p = 0$ (a positive probability of unsolvable episodes under the given generator/verifier

interface), then $\mathbb{E}[\mathrm{Err}_N(k)]$ converges to that atom mass and no amount of deliberation can drive the error to 0. The regime (6) lies between these extremes and is the one in which inference-time scaling laws are both nontrivial and informative.

The next section complements (11) by showing that the exponent a_N is not merely an artifact of the particular "sample-until-accept" rule: under the candidate-and-verify access model, no algorithm restricted to k samples can improve the asymptotic exponent on worst-case distributions satisfying (6).

5 Tight Bounds and Minimax Optimality

We now complement the asymptotic upper bound implicit in (11) by showing that, under the candidate-and-verify access pattern, the exponent a_N is minimax-optimal: no procedure restricted to k candidate generations and verifications can improve the worst-case scaling over episode distributions whose density near 0 satisfies (6). We also record several robustness statements, clarifying which modeling perturbations preserve the same compute–reliability law and which fall outside the oracle model.

A minimax formulation. Fix N and consider an arbitrary (possibly adaptive, randomized) algorithm A that, on an episode τ, may sequentially query the sampling oracle to obtain candidates $y_1, y_2, \ldots$ and the verification oracle to obtain outcomes $s_i \in \{0,1\}$, stopping after at most k queries in total. Under (H1)–(H2) with i.i.d. candidates, conditional on τ the verification outcomes are i.i.d. Bernoulli($P_{\tau,N}$) (after absorbing any benign failure modes into $P_{\tau,N}$). Hence A observes only a stream of failures until (possibly) the first acceptance, at which point it may stop and succeed. In particular, for fixed $p \in [0,1]$, the success event under any A using at most k samples is contained in the event that at least one of k independent Bernoulli(p) trials equals 1. Therefore, writing $\mathrm{Succ}_p(\mathsf{A}, k)$ for the success indicator when $P_{\tau,N} = p$ deterministically,

$$\mathbb{P}(\mathrm{Succ}_p(\mathsf{A},k)=1) \;\le\; 1-(1-p)^k, \qquad \mathbb{P}(\mathrm{Err}_p(\mathsf{A},k)=1) \;\ge\; (1-p)^k. \tag{14}$$

The inequality (14) is tight for the canonical "sample-until-accept" rule, but the point is that adaptivity cannot improve the dependence on p without additional structure: failures convey no actionable information beyond the fact that success has not yet occurred.

Matching lower bounds via Yao. To pass from (14) to a minimax statement over episode distributions, we apply Yao's principle: it suffices to exhibit a randomized instance distribution over P such that every deterministic k-query algorithm has expected error at least of order k^{-a_N}. Let F be any

distribution on $[0,1]$ with density satisfying (6). Then for every algorithm A,

$$\mathbb{E}_{P\sim F}[\mathbb{P}(\mathrm{Err}_P(\mathsf{A},k)=1\,|\,P)] \;\geq\; \mathbb{E}_{P\sim F}\Big[(1-P)^k\Big] \;=\; \int_0^1 (1-p)^k f_N(p)\,dp. \tag{15}$$

By the same Laplace-type asymptotics used previously (now interpreted as a lower bound because (14) holds for all A), we obtain

$$\inf_{\mathsf{A}:\,\leq k \text{ queries}} \sup_{F:\, f_N(p)=c_N p^{a_N-1}(1+o(1)) \text{ near } 0} \mathbb{E}[\mathrm{Err}_N(k)] \;\geq\; c_N\Gamma(a_N)\,k^{-a_N}\,(1+o(1)). \tag{16}$$

Together with (11), this yields matching upper and lower constants for a natural class of distributions whose small-p behavior is controlled by (a_N, c_N), establishing that the power-law exponent is information-theoretically sharp under the oracle model. In particular, any improvement in the exponent must come from relaxing the access model (e.g., allowing $S_N(\cdot\mid\tau)$ to condition on past outcomes in a way that increases $P_{\tau,N}$) rather than from a more sophisticated stopping rule.

Robustness to imperfect verification. Assumption (H2) can be weakened without changing the scaling exponent. Suppose instead that verification is noisy: given a candidate, the verifier outputs $s\in\{0,1\}$ with false-negative rate β_τ and false-positive rate α_τ, and executing an accepted candidate succeeds with probability at least $1-\alpha_\tau$ (so false positives correspond to unsafe or infeasible plans). Conditional on τ, the probability that a single generate–verify–execute attempt both passes verification and succeeds is

$$Q_{\tau,N} \;:=\; \mathbb{P}(\text{accept \& succeed}\mid\tau,N) \;=\; (1-\beta_\tau)P_{\tau,N}\cdot(1-\alpha_\tau), \tag{17}$$

under the natural conditional independence idealization. If α_τ,β_τ are bounded away from 1 and do not introduce additional heavy tails near 0, then $Q_{\tau,N}$ inherits the same tail exponent as $P_{\tau,N}$: for small p, the map $p\mapsto q=\eta_\tau p$ with $\eta_\tau=(1-\beta_\tau)(1-\alpha_\tau)$ is locally linear, hence the density of Q near 0 still behaves as $\tilde{c}_N q^{a_N-1}$. Consequently,

$$\mathbb{E}[\mathrm{Err}_N(k)] \;=\; \mathbb{E}\Big[(1-Q_{\tau,N})^k\Big] \;=\; \Theta(k^{-a_N}), \tag{18}$$

with constants rescaled by the distribution of η_τ. If, however, η_τ itself has substantial mass near 0 (e.g., the verifier is frequently uninformative), then the effective tail exponent can decrease; this corresponds to deployment regimes in which verification noise creates additional "nearly-impossible" episodes.

Robustness to dependence across candidates. The i.i.d. assumption (H1) is likewise relaxable to weak dependence. Let E_i denote the event that the ith candidate both passes verification and succeeds when executed, and write $q_\tau := \mathbb{P}(E_i \mid \tau)$ for the marginal per-attempt success probability. If, conditional on τ, the events $\{E_i\}$ are negatively associated or satisfy an exponential-mixing condition implying an upper bound of the form

$$\mathbb{P}\left(\bigcap_{i=1}^{k} E_i^c \,\middle|\, \tau\right) \;\le\; \exp(-\kappa k q_\tau) \qquad \text{for some } \kappa \in (0,1], \tag{19}$$

then the same Laplace analysis applies with an effective compute κk, and the expected error remains $\Theta(k^{-a_N})$ whenever the density of q_τ near 0 behaves as q^{a_N-1}. Conversely, strong positive dependence can reduce the effective number of independent trials (e.g., repeated sampling of near-duplicates), which changes constants and may even destroy the power law if it induces an atom at $q_\tau = 0$. This delineates a concrete operational requirement: candidate generation must explore sufficiently diverse plans for additional compute to translate into additional independent chances of success.

Extensions to imperfect critics and world models. Finally, many practical systems employ graded critics, learned feasibility scores, or world-model rollouts rather than a perfectly sound accept/reject verifier. In our abstraction, such mechanisms amount to replacing the binary oracle by a stochastic filter and then choosing an execution rule (e.g., execute the highest-scoring candidate). Under minimal assumptions—namely that, conditional on τ, each candidate produces an i.i.d. score s and an execution outcome, and the algorithm's decision is measurable with respect to observed scores—the probability of eventual success after k candidates is still upper bounded by the probability that at least one candidate is truly executable. Hence the minimax lower bound persists unless the additional signals allow the algorithm to *increase* the underlying per-candidate executability probability through structured resampling or proposal adaptation. When world-model rollouts are used solely as a verifier surrogate, their errors enter exactly as in (17). When, instead, rollouts are used to optimize candidates (e.g., search in plan space using simulated gradients), the effective $P_{\tau,N}$ becomes a function of k rather than a fixed latent parameter, placing the method outside (H1) and making genuinely better exponents possible in principle.

These tightness and robustness results justify treating a_N as the canonical descriptor of inference-time compute scaling under candidate-and-verify access. In the next section we compare the marginal benefits of increasing k versus increasing N, and we formalize when a phase transition in optimal resource allocation must occur.

6 Phase Transition Between Model Scaling and Inference Scaling

We now formalize when additional test-time compute k is the dominant lever for improving expected success, versus enlarging the deployed model size N (via additional training) while holding test-time compute fixed. Throughout we fix an episode horizon H and the candidate-and-verify access model, and we interpret N as fixed during deployment.

Asymptotic error model and marginal returns. Write

$$S(N,k) \;:=\; \mathbb{E}[\mathrm{Succ}_N(k)] \;=\; 1-\mathbb{E}[\mathrm{Err}_N(k)]\,.$$

Under (H1)–(H3), Theorem 1 yields the asymptotic form

$$\mathbb{E}[\mathrm{Err}_N(k)] \;=\; A_N\,k^{-a_N}\,(1+o(1)), \qquad A_N \;:=\; c_N\Gamma(a_N), \tag{20}$$

hence

$$S(N,k) \;=\; 1-A_N k^{-a_N}+o\big(k^{-a_N}\big)\,. \tag{21}$$

We compare marginal gains in expected success from increasing k versus increasing N. For test-time compute, differentiating (21) (formally treating k as continuous) gives

$$M_{\mathrm{test}}(N,k) \;:=\; \frac{\partial S}{\partial k}(N,k) \;=\; a_N A_N\,k^{-a_N-1}\,(1+o(1)). \tag{22}$$

For model size, it is natural to measure marginal return per multiplicative increase in N, i.e. per unit of $\log N$:

$$M_{\mathrm{model}}(N,k) \;:=\; \frac{\partial S}{\partial \log N}(N,k) \;=\; -\frac{\partial}{\partial \log N}\big(A_N k^{-a_N}\big)+o\big(k^{-a_N}\big)\,. \tag{23}$$

Expanding the derivative and using $A_N = c_N\Gamma(a_N)$, we obtain

$$M_{\mathrm{model}}(N,k) \;=\; k^{-a_N}\left[A_N(\log k)\,\frac{\partial a_N}{\partial \log N}-\frac{\partial A_N}{\partial \log N}\right]+o\big(k^{-a_N}\big)\,, \tag{24}$$

with

$$\frac{\partial A_N}{\partial \log N} \;=\; \Gamma(a_N)\,\frac{\partial c_N}{\partial \log N}+A_N\,\psi(a_N)\,\frac{\partial a_N}{\partial \log N}, \tag{25}$$

where $\psi=\Gamma'/\Gamma$ is the digamma function.

A crossing condition and existence of a threshold N^*. We say that inference-time scaling dominates at (N, k) if $M_{\text{test}}(N, k) > M_{\text{model}}(N, k)$, i.e. one additional unit of test-time compute yields more expected success than a marginal multiplicative increase in model size. Using (22)–(24), and ignoring lower-order terms in k, the dominance condition is

$$a_N A_N \, k^{-a_N - 1} \;>\; k^{-a_N} \left[A_N (\log k) \, \frac{\partial a_N}{\partial \log N} - \frac{\partial A_N}{\partial \log N} \right], \tag{26}$$

or equivalently

$$\frac{a_N}{k} \;>\; (\log k) \, \frac{\partial a_N}{\partial \log N} - \frac{1}{A_N} \frac{\partial A_N}{\partial \log N}. \tag{27}$$

This makes explicit how a phase transition can occur. If, as N grows, improvements in the tail parameters saturate in the sense that

$$\frac{\partial a_N}{\partial \log N} \to 0, \qquad \frac{\partial \log A_N}{\partial \log N} \to 0, \tag{28}$$

then the right-hand side of (27) tends to 0, while the left-hand side remains $a_N/k > 0$ at fixed k. Under mild regularity (e.g. continuity in N and eventual monotonic decrease of the right-hand side), there exists a (possibly k-dependent) $N^*(k)$ such that for all $N \geq N^*(k)$, inference-time compute has the larger marginal return. Conversely, if $\partial a_N / \partial \log N$ remains bounded away from 0, then the term $A_N (\log k) \, \partial a_N / \partial \log N$ can dominate for large k, and enlarging N may remain competitive even at substantial test-time compute.

Latency constraints and feasibility of target error. A practically relevant comparison fixes a latency cap K and asks whether compute alone can reach a target expected error ϵ. Using (20), achieving $\mathbb{E}[\text{Err}_N(k)] \leq \epsilon$ requires

$$k \;\gtrsim\; \left(\frac{A_N}{\epsilon} \right)^{1/a_N}. \tag{29}$$

If $K < (A_N/\epsilon)^{1/a_N}$, then no compute-allocation strategy respecting the per-episode latency cap can reach ϵ with model size N within our oracle abstraction; the only remaining lever is to increase N so as to decrease A_N and/or increase a_N until

$$A_N \, K^{-a_N} \;\lesssim\; \epsilon. \tag{30}$$

Thus, under strict latency, model scaling is forced whenever (30) fails; whereas when (30) holds with margin, one expects diminishing returns from further model scaling and increasing k up to K becomes the direct path to reliability improvements.

Resource-allocation guidance under a compute penalty. If we introduce a per-episode utility $U(N,k) = S(N,k) - \lambda k$ with $\lambda > 0$, then (22) shows that the locally optimal k at fixed N satisfies (again at the level of the asymptotics)

$$a_N A_N \, k^{-a_N - 1} \;\approx\; \lambda, \qquad \text{i.e.} \qquad k^*(N) \;\approx\; \left(\frac{a_N A_N}{\lambda}\right)^{\frac{1}{a_N+1}}, \tag{31}$$

subject to truncation by K. When N is large enough that (28) holds, $k^*(N)$ changes slowly with N, and improvements in U are primarily obtained by adjusting k (or, under a global budget, by reallocating k_τ across episodes). When N is small and $\partial a_N / \partial \log N$ is substantial, the term $A_N (\log k)\, \partial a_N / \partial \log N$ in (24) indicates that increasing N can be particularly valuable in regimes where k is already moderately large, because improvements in a_N amplify the effect of every additional test-time sample through the factor k^{-a_N}.

The preceding comparison is pointwise in (N,k) and does not yet prescribe how to choose k when $P_{\tau,N}$ varies across episodes and is not observed directly. We turn next to compute controllers that allocate deliberation adaptively across episodes and within episodes, obtaining approximation guarantees offline and regret bounds online.

7 Adaptive Compute Allocation Algorithms

We now specify compute controllers that (a) allocate a global budget across episodes and (b) optionally stop early within an episode when additional sampling is not worth its opportunity cost. The main difficulty is that $P_{\tau,N}$ is latent and episode-specific; the controller only observes verifier outputs (accept/reject or scores) and possibly cheap side information $z = z(\tau)$.

Offline budgeted allocation (known or estimated difficulty). Consider first an offline benchmark in which we are given T episodes $\tau_1, \ldots, \tau_T$ together with their latent parameters $P_t := P_{\tau_t,N}$, and a global budget $\sum_{t=1}^{T} k_t \le B$ with $k_t \in \{0, \ldots, K\}$. Under the i.i.d. candidate model and sound verification, the expected success on episode t is

$$g_{P_t}(k_t) \;=\; 1 - (1 - P_t)^{k_t}.$$

By Theorem 3, each g_{P_t} is concave (diminishing returns), and by Theorem 4 the greedy policy that repeatedly assigns the next compute unit to the episode with the largest marginal gain

$$\Delta_t(j) \;:=\; g_{P_t}(j+1) - g_{P_t}(j) \;=\; P_t(1-P_t)^j$$

achieves a $(1-1/e)$-approximation to the offline optimum under a cardinality budget; the same conclusion holds with weighted per-sample costs by allocating according to $\Delta_t(j)/w$ in the standard knapsack-submodular variants.

This provides a principled target for any implementable controller: we seek to approximate the greedy marginal-gain ordering without direct access to P_t.

In practice P_t must be replaced by an estimator $\widehat{P}_t$. A useful stability fact is that $g_p(k)$ is Lipschitz in p:

$$\left|g_p(k) - g_q(k)\right| \;=\; \left|(1-q)^k - (1-p)^k\right| \;\le\; k|p-q|, \tag{32}$$

since the derivative in p is bounded by k on $[0,1]$. Thus plug-in allocation based on $\widehat{P}_t$ incurs at most an additive distortion controlled by $k_t|\widehat{P}_t - P_t|$ per episode, which justifies spending a small amount of compute on difficulty estimation when k_t is large.

Within-episode adaptive deliberation as optimal stopping. Within a single episode τ, we may decide after i failed candidates whether to draw candidate $i{+}1$ or to stop and execute a fallback. Without any compute penalty and without a global budget coupling across episodes, there is no reason to stop before the cap K, since the success probability $1-(1-P)^k$ is increasing in k. Early stopping becomes nontrivial in two settings: (i) we impose a per-unit penalty $\lambda > 0$ (as in a utility $S - \lambda k$), or (ii) we have a global budget and interpret one more within-episode query as consuming a scarce resource with an implicit shadow price λ.

A convenient formalization is Bayesian. Place a prior $P \sim \mathrm{Beta}(\alpha, \beta)$ for the episode's latent success probability, and update after failures (rejects) using conjugacy. Conditioning on i failures and no accept so far, we have the posterior

$$P \,\big|\, (i \text{ failures}) \;\sim\; \mathrm{Beta}(\alpha, \beta + i), \qquad \mathbb{E}[P \mid i] \;=\; \frac{\alpha}{\alpha + \beta + i}.$$

If we value success at 1 and charge cost λ per additional sample, then the myopic one-step improvement from drawing one more candidate is exactly the posterior mean $\mathbb{E}[P \mid i]$ (since the next draw succeeds with probability P and otherwise leaves us where we started). Hence a simple stopping rule is the threshold test

$$\text{continue at step } i{+}1 \text{ iff } \quad \frac{\alpha}{\alpha + \beta + i} \;>\; \lambda, \tag{33}$$

subject to the hard cap K. When a global budget is present, we take λ to be the current estimate of the opportunity cost of one unit of compute (equivalently, a Lagrange multiplier chosen to approximately saturate $\sum_t k_t \le B$). More refined stopping rules can incorporate score-valued verifiers by replacing the Bernoulli observation model with a calibrated likelihood for $s_i = V(y_i, \tau)$ and maintaining a posterior over P; the same comparison of marginal value to λ applies, with $\mathbb{E}[P \mid \text{data}]$ computed under the score model.

Online allocation under a global budget (bandits with knapsacks). We next consider the streaming setting in which episodes $\tau_1, \ldots, \tau_T$ arrive sequentially and we must decide k_t online while respecting $\sum_{t=1}^{T} k_t \leq B$. Here P_t is unobserved and varies across episodes; the controller only sees a context z_t (cheap difficulty features) and sampling outcomes. This can be cast as a contextual bandits-with-knapsacks problem: choosing an "action" $k \in \{0, \ldots, K\}$ consumes cost k and yields reward $g_{P_t}(k)$, with P_t drawn from an unknown conditional distribution given z_t.

General BwK theory implies an unavoidable exploration cost: any algorithm must sometimes spend compute suboptimally to learn which contexts warrant deliberation. Conversely, standard optimistic (UCB-style) or posterior-sampling (Thompson-style) controllers achieve sublinear regret under mild assumptions (bounded rewards, bounded costs, and either finite action sets or controlled function classes for mapping z to predicted difficulty). Concretely, after discretizing the action set to $\{0, \ldots, K\}$, one obtains regret bounds of the form $\widetilde{O}(\sqrt{T(K+1)})$ (or budget-dependent analogues) against the best fixed policy in hindsight, with the precise rate depending on the chosen model class for context-to-reward prediction and on whether one competes with stationary or adaptive benchmarks.

Practical estimators of per-episode difficulty. Finally we record estimators $\widehat{P}_\tau$ suitable for driving either greedy offline allocation or online BwK controllers. The simplest is a *probe-and-allocate* scheme: spend a small initial budget k_0 on each episode, compute an empirical acceptance rate $\widehat{P} = (s+\alpha)/(k_0+\alpha+\beta)$ with a Beta prior, and then allocate the remaining budget using the greedy marginal rule with P replaced by $\widehat{P}$. When verifier outputs are score-valued, we may map scores to calibrated probabilities (e.g. via isotonic regression on held-out data) and treat the calibrated score as an estimate of $\mathbb{E}[P \mid s]$. In embodied settings, additional low-cost signals $z(\tau)$ (model uncertainty, disagreement in an ensemble critic, heuristic measures of scene clutter, or language-model self-reported uncertainty) can be regressed to predict $\widehat{P}$ before any sampling; these predictors are most valuable when B/T is small and probing every episode is expensive. In all cases, the algorithmic objective is the same: approximate the greedy ordering of marginal gains $\Delta_\tau(j)$, while using within-episode stopping rules such as (33) to avoid wasting compute on episodes whose posterior difficulty is so high that the expected return per unit compute falls below the current shadow price.

8 Experimental Protocol (Optional but Strengthening)

We describe an experimental protocol that makes the assumptions and predictions of the candidate-and-verify model directly testable using an open-

source vision–language–action (VLA) policy as π_N, together with an explicit sampler $S_N(\cdot \mid \tau)$, a verifier $V(\cdot, \tau)$, and (optionally) a learned world model used only at test time. The goal is not to optimize absolute performance, but to (i) sweep test-time compute C_{test} in a controlled manner, (ii) estimate the inference scaling exponent a_N in $\mathbb{E}[\text{Err}_N(k)] \asymp k^{-a_N}$, (iii) measure how (a_N, c_N) vary with N, and (iv) identify an empirical threshold N^* at which additional test-time compute yields larger marginal gains than increasing N.

Wrapping a VLA as a candidate generator. Fix an environment suite defining $\mathcal{T}$ (e.g. tabletop manipulation with language instructions, navigation-and-pick, or multi-step tool use), and fix a standardized episode representation τ containing the instruction, initial observation x_0, and any allowed scene metadata. We implement a *plan* y as either (a) an open-loop sequence of low-level actions of length H, (b) a short sequence of discrete skills with parameters, or (c) a policy prompt (for hierarchical prompting) that induces a closed-loop rollout when executed. The sampler $S_N(\cdot \mid \tau)$ is induced by π_N by introducing explicit stochasticity: temperature or nucleus sampling for token-based action plans; diffusion noise for continuous trajectories; or randomized decoding (top-k sampling, stochastic beam) for skill sequences. We record the exact sampling hyperparameters because they affect both the independence approximation and the tail of $P_{\tau,N}$; to reduce confounds we keep the sampler fixed across k-sweeps and across model sizes unless explicitly ablated.

Verifier and soundness checks. We instantiate $V(y, \tau)$ as either (i) a learned critic (value function, success classifier, or constraint classifier), (ii) a rule-based feasibility checker (collision, kinematic reachability, safety envelopes), or (iii) an ensemble that returns accept iff all components accept. Since our analysis treats verification as sound up to a bounded false-positive rate, we empirically bound this rate by auditing a random subset of accepted candidates: execute them in the real/sim environment and estimate $\widehat{\text{FP}} = \mathbb{P}(\text{fail} \mid \text{accept})$. If $\widehat{\text{FP}}$ is non-negligible, we report results both (a) with raw acceptance, and (b) with a tightened accept threshold chosen to reduce $\widehat{\text{FP}}$ to a target level, noting that tightening typically reduces $P_{\tau,N}$ and thus changes c_N.

Optional world-model rollouts. To test the effect of additional structure beyond i.i.d. candidate draws, we optionally augment V with a world-model rollout: for each candidate y, simulate predicted execution and accept only if the simulation terminates in a predicted success state with high probability. We account for the added cost by using weighted compute units: if generation, verification, and simulation costs are (w_g, w_v, w_s), then the per-candidate cost is $w = w_g + w_v + w_s$ and the budget is a constraint

$\sum w \le C_{\text{test}}$. When reporting scaling in k, we additionally report scaling in the wall-clock-normalized budget C_{test} to avoid artifacts from expensive verifiers.

Controlled compute sweeps. For each fixed N, we evaluate success as a function of the per-episode cap K by running the same episode set multiple times with different budgets $k \in \{0, 1, \ldots, K_{\max}\}$. The primary measurement is

$$\widehat{\mathrm{Err}}_N(k) \;=\; 1 - \frac{1}{M}\sum_{m=1}^{M} \mathrm{Succ}_{\tau_m,N}(k),$$

where $\tau_1, \ldots, \tau_M \sim \mathcal{T}$ are held fixed across budgets to reduce variance. We then fit an exponent $\widehat{a}_N$ by regressing $\log \widehat{\mathrm{Err}}_N(k)$ on $\log k$ over a range $k \in [k_{\min}, k_{\max}]$ chosen to avoid trivial regimes (very small k dominated by non-asymptotic effects, and very large k near saturation). We report (i) the fitted slope with confidence intervals via bootstrap over episodes, and (ii) goodness-of-fit diagnostics. Because finite-sample effects can mimic power laws, we also report a two-parameter fit $\widehat{\mathrm{Err}}_N(k) \approx Ak^{-\widehat{a}_N} + b$ to detect an error floor $b > 0$ (which corresponds to tasks with effectively $P_{\tau,N} = 0$ under the fixed sampler/verifier).

Estimating the tail parameters (a_N, c_N)**.** Beyond fitting $\widehat{\mathrm{Err}}_N(k)$, we can estimate the near-zero behavior of the latent $P_{\tau,N}$. We approximate $P_{\tau,N}$ by repeated probing: for each episode τ, draw k_0 candidates and compute the empirical acceptance indicator sequence. Under the Bernoulli model, a Beta posterior yields $\widehat{P}_\tau = \frac{\alpha+s}{\alpha+\beta+k_0}$, where s is the number of accepts. Aggregating $\{\widehat{P}_\tau\}$ across episodes, we fit a density near 0 by log-log regression of the empirical CDF: for small p,

$$\widehat{F}_N(p) \;=\; \mathbb{P}(\widehat{P}_\tau \le p) \;\approx\; \widetilde{c}_N p^{\widehat{a}_N},$$

which gives an independent estimate of $\widehat{a}_N$ that should be consistent with the error-scaling estimate if the stylized assumptions are approximately valid.

Identifying an empirical threshold N^***.** To compare scaling in N versus scaling in k, we train or select a family of models $\{\pi_N\}$ (e.g. different parameter counts, or different fine-tuning budgets) and repeat the compute sweep for each N. For a fixed operating point (N, k) we estimate the marginal test-time gain by a finite difference

$$\widehat{M}_{\text{test}}(N,k) \;=\; \widehat{S}(N,k+1) - \widehat{S}(N,k),$$

and the marginal model-size gain by

$$\widehat{M}_{\text{model}}(N,k) \;=\; \frac{\widehat{S}(N_2,k) - \widehat{S}(N_1,k)}{\log N_2 - \log N_1},$$

for adjacent sizes $N_1 < N_2$ around N. We then define $\widehat{N}^*(k)$ as the smallest N such that $\widehat{M}_{\text{test}}(N,k)$ exceeds $\widehat{M}_{\text{model}}(N,k)$ within statistical error bars. In deployments with a fixed training budget, we additionally normalize by measured training cost to compare resource-to-resource tradeoffs.

Ablations and confound controls. To support causal interpretation, we recommend ablations that isolate deviations from (H1)–(H3). (i) *Correlation ablation:* vary decoding randomness (temperature, top-p) while holding k fixed to check whether diversity increases effective independence and steepens scaling; report an estimated effective sample size via acceptance autocorrelation across candidates. (ii) *Verifier ablation:* swap V among rule-based, learned, and ensemble verifiers to test sensitivity of (a_N, c_N) to soundness and calibration; separately report $\widehat{\text{FP}}$ and $\widehat{\text{FN}}$. (iii) *World-model ablation:* toggle simulation on/off and report scaling as a function of the weighted budget C_{test}; this distinguishes genuine inference-time scaling from merely spending more expensive compute per candidate. (iv) *Nonstationarity control:* freeze environment seeds and initial conditions across budgets and sizes, and separately evaluate on a time-shifted task distribution to test robustness of fitted exponents. (v) *Fallback control:* fix the fallback policy and report its standalone success, since a strong fallback can mask scaling at small k.

All reported results should include the full compute accounting (number of candidates, verifier calls, simulator steps, and wall-clock), the fitted scaling range, and uncertainty estimates over $\tau \sim \mathcal{T}$. This renders the exponent claims falsifiable and makes the location of any empirical "phase transition" N^* comparable across systems.

9 Discussion and Limitations

Our analysis isolates a stylized *candidate-and-verify* mechanism and derives a power-law test-time scaling $\mathbb{E}[\text{Err}_N(k)] \asymp k^{-a_N}$ under a heavy-tail condition on the latent per-episode success probability $P_{\tau,N}$. We now delineate regimes in which this prediction may fail or become operationally misleading, and we articulate implications for safety and benchmarking practice.

When power laws may not appear. Theorem 1 is an asymptotic statement driven by the behavior of $f_N(p)$ near $p = 0$. Consequently, several departures are expected.

(i) Saturation and error floors. If there is nontrivial mass at $P_{\tau,N} = 0$ (or, more generally, if $P_{\tau,N}$ is so small that it is effectively zero within the allowed cap K), then $\mathbb{E}[(1 - P_{\tau,N})^k]$ converges to a positive constant as $k \to \infty$, and a pure power law must eventually break. Empirically this manifests as an error floor $b > 0$ in fits of the form $\widehat{\text{Err}}_N(k) \approx Ak^{-\widehat{a}_N} + b$. In robotics, such floors can correspond to irrecoverable perceptual failures, missing tools, or

instruction ambiguity that cannot be resolved by additional sampling from a fixed $S_N(\cdot \mid \tau)$ and fixed verifier $V(\cdot, \tau)$. We therefore interpret power-law fits as claims about an intermediate regime: after initial transients and before saturation.

(ii) Non-heavy-tail episode distributions. If $P_{\tau,N}$ is bounded away from 0 with high probability (e.g. $P_{\tau,N} \geq p_{\min} > 0$), then $\mathbb{E}[\mathrm{Err}_N(k)] \leq (1 - p_{\min})^k$ decays exponentially rather than polynomially. Conversely, if $f_N(p)$ is heavier than the assumed regular variation near 0 (e.g. slowly varying factors, or mixtures with multiple scales), then a single exponent a_N may not describe the curve over any substantial range of k. This is not a defect of the proof but a reminder that the exponent is a property of the task distribution *as seen through* the chosen sampler and verifier.

Correlated candidates and the effective compute budget. Assumption (H1) posits conditional independence across candidates. In practice, stochastic decoding often produces highly correlated plans: diverse prefixes collapse to similar suffixes, diffusion samples concentrate on a few modes, and implicit replanning can repeatedly revisit the same local minimum. Under dependence, the exact success probability is no longer $1 - (1 - P_{\tau,N})^k$, and marginal gains can be substantially smaller than predicted. One convenient way to summarize this effect is via an *effective sample size* $k_{\mathrm{eff}}(k) \leq k$ such that

$$\mathbb{P}(\text{no success in } k \text{ draws} \mid \tau) \;\approx\; (1 - P_{\tau,N})^{k_{\mathrm{eff}}(k)}.$$

Correlations typically induce $k_{\mathrm{eff}}(k)$ that grows sublinearly in k over relevant ranges, flattening log-log slopes and creating apparent "deviations from power laws" that are in fact deviations from independence. This limitation is actionable: reporting diversity controls (temperature, top-p, noise scale) and measuring acceptance autocorrelation across candidates clarifies whether improvements come from more compute or from a higher k_{eff} per unit compute.

Verifier imperfections and safety. We have treated verification as sound up to a bounded false-positive rate absorbed into $P_{\tau,N}$. In safety-critical deployments this abstraction can hide a qualitatively different failure mode: increasing k can increase the probability of encountering a *false* accept even when true accepts are rare. Concretely, if the verifier accepts an unsafe plan with probability $\alpha > 0$ per candidate independently of true feasibility, then the probability of at least one unsafe accept scales as $1 - (1 - \alpha)^k$, which increases with k and can dominate any success gains. Thus, the same mechanism that yields inference scaling can amplify risk unless verification is calibrated with respect to the deployment distribution and audited at the operating k. A conservative recommendation is to separate metrics: report

(i) task success conditional on accept, (ii) false accept rates (safety violations) as a function of k, and (iii) the joint utility $\mathbb{E}[\text{Succ}] - \lambda k$ for explicitly stated λ when comparing systems.

Nonstationarity and distribution shift. Assumption (H3) fixes an episode distribution $\mathcal{T}$ and hence a fixed f_N. In embodied settings, however, the effective distribution can drift: sensors degrade, clutter varies, human instructions change, or the agent itself alters the environment over time. Under such nonstationarity, a controller tuned to a single exponent a_N may allocate compute poorly: what was once a "hard tail" episode might become common, or vice versa. Moreover, global-budget allocation becomes an online learning problem akin to bandits with knapsacks: one must spend compute to infer difficulty while preserving budget for exploitation. In this regime, we should expect regret-type tradeoffs rather than static approximation guarantees, and we should evaluate compute controllers on time-ordered streams rather than i.i.d. episode sets.

Benchmarking implications and reporting recommendations. Because scaling claims are sensitive to sampling, verification, and accounting conventions, we recommend the following as a minimal reporting standard for inference-compute studies.

1. *Compute units and costs:* specify what constitutes one unit of k (candidate generation, verification, simulation), and report weighted budgets when costs differ; include wall-clock latency distributions, not only counts.

2. *Sampler specification:* report decoding/noise hyperparameters and any diversity mechanisms; if multiple candidates are generated in a batched or beam-like manner, clarify the dependence structure.

3. *Verifier operating point:* report acceptance thresholds, calibration procedure, and estimated false-positive/false-negative rates on audited subsets; include safety violation rates versus k.

4. *Curves, not single points:* publish $\widehat{\text{Err}}_N(k)$ (and success) over a range of k with confidence intervals; state the fitted range $[k_{\min}, k_{\max}]$ and include alternative fits allowing an error floor.

5. *Reproducibility controls:* fix episode sets across k-sweeps, disclose random seeds for candidate sampling, and report fallback policy performance separately.

These practices make it possible to interpret an observed exponent as a property of the deployed inference procedure rather than an artifact of hidden compute, hidden correlations, or shifting safety thresholds.

10 Conclusion: takeaways and open problems

We have formalized a test-time *candidate-and-verify* abstraction for embodied agents and used it to relate inference compute to reliability through the latent per-episode success probability $P_{\tau,N}$. The central consequence is that, under a mild regular-variation hypothesis on the density of $P_{\tau,N}$ near 0, the expected error as a function of test-time budget obeys a power law, with an exponent a_N that is both identifiable from the instance distribution (as mediated by the sampler and verifier) and minimax-optimal in the sense that no k-limited procedure can improve the exponent without exploiting additional structure. In parallel, we have emphasized that per-episode success is concave in the number of verified candidates, which yields diminishing returns and allows principled budget allocation across episodes under global constraints.

From an engineering viewpoint, the immediate takeaway is that *inference-time compute is an algorithmic resource of the same status as model size.* The relevant performance object is not a single curve in k at fixed N, nor a single curve in N at fixed k, but a two-dimensional surface

$$S(N,k) \;=\; \mathbb{E}_{\tau\sim\mathcal{T}}\Big[1-(1-P_{\tau,N})^k\Big], \qquad \mathbb{E}[\mathrm{Err}_N(k)] \;=\; 1-S(N,k),$$

together with its derivatives that quantify marginal returns per unit resource. In embodied robotics this perspective is operationally necessary: latency caps K are hard, energy and thermal limits induce time-varying budgets, and the dominant failure modes often reside in the small-$P_{\tau,N}$ tail where additional sampling is most valuable. Theorem 5 supplies a precise way to articulate a "phase transition" between scaling regimes: when improvements in the tail parameters (a_N, c_N) saturate with training, increasing k can dominate further increases in N at deployment. Conversely, when the tail improves rapidly with N, training compute may remain the more effective lever. A practical implication is that benchmarking should report not only aggregate success at a single operating point, but also *resource-sensitivity*: how quickly success improves with k and how the implied tail parameters change with N.

Several open problems must be resolved to turn this abstraction into a predictive theory for robotics deployments.

(1) Joint train–test scaling laws as surfaces. Our results treat N as fixed at deployment and analyze scaling in k. A natural extension is to seek empirical and theoretical laws for $\mathbb{E}[\mathrm{Err}_N(k)]$ as a *joint* function of training resources and test-time compute. Concretely, one may posit parametric forms such as

$$\mathbb{E}[\mathrm{Err}_N(k)] \approx c_N \Gamma(a_N) k^{-a_N} + b_N,$$

with (a_N, c_N, b_N) depending smoothly on training compute, data, and architecture, and then ask for conditions under which these dependencies are stable across task families. In robotics, where the episode distribution is shaped by morphology, sensors, and environment design, it is not obvious that a_N should be monotone in N; additional capacity could change the *mode structure* of $S_N(\cdot \mid \tau)$ and thereby alter the tail in nontrivial ways. A satisfactory account should explain when scaling manifests primarily through a_N (changing the hardness tail) versus through c_N (changing mass near 0 without changing the exponent), and how these interact with realistic caps K.

(2) Beyond i.i.d. candidates: diversity mechanisms as first-class design. Independence of candidate outcomes is the simplifying axis of the present analysis, but practical samplers are controlled by a small number of diversity parameters. This suggests treating the sampler not merely as S_N but as a family $S_{N,\theta}$ indexed by a diversity control θ (temperature, noise scale, guidance strength, prompt randomization, or explicit determinantal/coverage objectives). The question is then to characterize the induced *effective* tail behavior of $P_{\tau,N,\theta}$ and to optimize the triple (N, θ, k) under latency. Even when the marginal gain in k remains diminishing, changing θ can alter the concavity itself by changing candidate correlations. A principled treatment would connect diversity controls to an explicit dependence model and derive guarantees in terms of a measurable quantity (e.g. an acceptance autocorrelation time or a coupling bound).

(3) Verifier design under coupled success and safety objectives. In embodied settings, "verification" often bundles multiple functions: constraint checking, collision prediction, policy value estimation, and sometimes learned safety classifiers. The relevant objective is therefore multi-criterion: maximize task success while bounding safety violations and possibly energy use. This raises two problems. First, if the verifier provides a calibrated *score* rather than a binary accept/reject, we should incorporate score-based selection (execute the best-scoring candidate) and analyze the associated scaling with k under assumptions on score distributions. Second, when safety is a constraint rather than a penalty, the compute allocator must respect a per-episode risk budget; this resembles constrained optimal stopping and constrained submodular maximization, for which sharp approximation guarantees in the present setting are incomplete.

(4) Online allocation under nonstationarity. Global-budget control in deployment is intrinsically online: the controller observes partial information and must allocate compute sequentially. Under distribution shift, the exponent a_N itself may drift, and the controller must trade exploration (esti-

mating difficulty) against exploitation (spending compute where it yields immediate success). Establishing regret bounds for compute controllers in the candidate-and-verify model, especially when the verifier emits informative scores, is a concrete theoretical target. For robotics, one must additionally integrate real-time constraints: compute decisions must be made with small overhead and with robustness to delayed or noisy feedback.

(5) Standardized inference-compute accounting for robotics. Finally, the field requires standard accounting conventions for inference compute that are meaningful for embodied agents. We propose that any deployment report a *compute ledger* specifying (i) the unit costs for generation, verification, and any simulation or rollout; (ii) the per-episode cap K and the distribution of realized k under adaptive stopping; and (iii) the wall-clock latency distribution on target hardware. Because robotics pipelines often amortize perception and mapping across candidates, it is essential to declare what is shared and what scales with k. Without such standardization, comparisons across systems conflate algorithmic gains with unreported compute, and the fitted exponents become artifacts rather than properties of the inference procedure. Establishing community benchmarks with fixed ledgers, fixed episode sets, and agreed-upon reporting of success, safety, and latency as functions of k would make scaling claims falsifiable and would clarify when inference-time deliberation is the appropriate route to reliability.

In summary, we have provided a tractable mathematical lens on inference-time deliberation for embodied agents. The main opportunity is to elevate (N, k) co-design—together with sampler diversity and verifier calibration—to a principled discipline with standardized measurement. The main risk is to treat any single fitted power law as universal. The open problems above delineate what must be understood for the candidate-and-verify model to become predictive across real robotic deployments.