

Verifier Reliability Under Budgeted Code Generation: Error Covariance and Information-Theoretic Stopping Bounds

Cisco Caceres · 10 October 2026 · Working Paper

Abstract

Test-time compute scaling methods such as Best-of-N sampling, tree search, and sequential self-correction rely on automated verifiers to select correct completions and decide when to terminate search. Standard verification algorithms assume that distinct verifiers commit conditionally independent errors given program correctness. In this work, we demonstrate mathematically and empirically that this independence assumption fails in practice. When heterogeneous verifiers evaluate candidate code solutions, subtle edge cases, ambiguous specifications, and common algorithmic traps induce strong positive error covariance ($\sigma_{AB} = 0.0934$, Pearson $\rho = 0.7523$). We formalize the decision-theoretic value of verifier information, deriving an elementary upper bound on policy improvement $\Delta \leq \min\{1 - V_0, \frac{1}{T}, \frac{1}{\sqrt{I/2}}\}$ in terms of total variation distance T and conditional mutual information $I(W; Z | X)$. Under positive covariance, naive Best-of-N suffers from severe false-consensus breakdown, lagging behind simple test filters under low compute budgets. We introduce Verifier Error Covariance & Sequential Calibration Rules (VEC-SCR), a sequential stopping and selection policy that penalizes verifier consensus when public unit tests fail. Across a 5-tier budget sweep ($1.0B^*$ to $16.0B^*$) on a frozen 504-task frontier benchmark (4,032 candidate pool) drawn from LiveCodeBench and BigCodeBench, VEC-SCR achieves a 29.0% pass rate at $2.0B^*$ (outperforming naive Best-of-N at 26.8%) and scales to 70.0% at $16.0B^*$ (matching naive Best-of-N while eliminating premature search exhaustion and approaching the 71.8% oracle ceiling). Under held-out algorithmic distribution shift at $4.0B^*$, VEC-SCR demonstrates robust generalization (44.6% vs 44.0% for naive Best-of-N), conserving inference tokens while matching oracle headroom.

1. Introduction & Problem Formulation

Inference-time reasoning has emerged as a primary lever for expanding the capabilities of generative foundation models on formal reasoning tasks. Rather than relying solely on single-pass greedy decoding (pass@1), practitioners allocate additional test-time compute to sample K candidate solutions, evaluate them with automated verifiers, and select the highest-scoring candidate.

In automated software engineering and code generation, automated verification typically takes two forms:

1. **Static Execution Verifiers (V_A):** Execution of available public test suites, linting rules, syntax parsers, and invariant checks.
2. **Generative Evaluators (V_B):** Neural process reward models or prompt-based model judges evaluating semantic completeness, edge case coverage, and algorithmic efficiency.

Virtually all standard reranking and aggregation frameworks, including majority voting, consensus thresholding, and weighted product-of-experts scoring, rely on the assumption of conditional independence:

$$P(V_A = v_A, V_B = v_B | Y = y) = P(V_A = v_A | Y = y) \cdot P(V_B = v_B | Y = y)$$

where $Y \in \{0, 1\}$ denotes true latent program correctness.

When verifiers are conditionally independent, combining scores from M noisy verifiers drives aggregate false acceptance exponentially toward zero. However, in automated code generation, this assumption is fundamentally violated. A candidate solution that contains an off-by-one boundary defect on large inputs often passes public unit tests and presents a superficially clean, convincing implementation that fools a neural evaluator. Consequently, errors between heterogeneous verifiers exhibit strong positive covariance.

When positive error covariance is present, naive consensus mechanisms become deceptive: they amplify shared false positives, leading selectors to prefer subtly incorrect code over correct candidates. Additionally, spending compute tokens to acquire additional verifier signals yields diminishing or negative returns if the acquisition cost exceeds the information-theoretic value of the signal.

This working paper addresses two core questions:

1. **Information-Theoretic Ceiling:** What is the maximum decision-theoretic utility gain achievable from an imperfect verifier signal, and at what cost does acquiring verification become provably suboptimal?
2. **Empirical Policy Allocation:** How does positive error covariance manifest across heterogeneous verifiers on challenging coding benchmarks, and how can sequential stopping rules maintain accuracy while conserving test-time compute?

2. Information-Theoretic Stopping Bounds

We formalize candidate selection and verifier acquisition as a finite-action decision problem under partial observation.

2.1 FORMAL DECISION FRAMEWORK

Let $W \in \mathcal{W}$ denote a finite latent outcome state representing the joint correctness vector of candidate solutions $W = (Y_1, \dots, Y_K) \in \{0, 1\}^K$. Let $X \in \mathcal{X}$ denote the initial observation available to the agent, including the task prompt, problem specification, and public unit test results. Let $Z \in \mathcal{Z}$ denote an additional verifier observation, such as execution logs from supplementary test suites or scores from neural judges.

A finite action set $\mathcal{A}$ is available to the decision maker, consisting of selecting candidate $j \in \{1, \dots, K\}$ or abstaining ($a = \emptyset$). For each action $a \in \mathcal{A}$, let $u(a, x, w) \in [0, 1]$ represent the utility of action a given observation x and latent state w . For candidate selection, $u(j, x, w) = Y_j$ corresponds to binary correctness, while abstention yields $u(\emptyset, x, w) = 0$.

Let $p_X(w) = P(W = w \mid X = x)$ denote the prior belief over latent states given initial observation X . Let $q_{XZ}(w) = P(W = w \mid X = x, Z = z)$ denote the posterior belief updated after observing verifier signal Z .

Define the prior Bayes-optimal value V_0 and posterior Bayes-optimal value V_1 :

$$V_0 = \mathbb{E}_X[\max_{a \in \mathcal{A}} \sum_{w \in \mathcal{W}} p_X(w) u(a, X, w)]$$

$$V_1 = \mathbb{E}_{XZ}[\max_{a \in \mathcal{A}} \sum_{w \in \mathcal{W}} q_{XZ}(w) u(a, X, w)]$$

The net value of information provided by verifier signal Z is defined as $\Delta = V_1 - V_0$.

2.2 DERIVATION OF THE INFORMATION CEILING

We define the expected Total Variation (TV) distance between posterior and prior distributions:

$$T = \mathbb{E}_{XZ} [\text{TV}(q_{XZ}, p_X)] = \frac{1}{2} \cdot \mathbb{E}_{XZ} [\sum_{w \in \mathcal{W}} |q_{XZ}(w) - p_X(w)|]$$

and the conditional mutual information $I(W; Z | X)$ measured in nats:

$$I = I(W; Z | X) = \mathbb{E}_{XZ} [D_{\text{KL}}(q_{XZ} \parallel p_X)] = \mathbb{E}_{XZ} [\sum_{w \in \mathcal{W}} q_{XZ}(w) \ln(q_{XZ}(w) / p_X(w))]$$

Theorem 1 (Information-Theoretic Ceiling on Verifier Gain). For any finite action set $\mathcal{A}$, bounded utility $u \in [0, 1]$, and joint distribution $P(W, X, Z)$:

$$0 \leq \Delta \leq \min\{ 1 - V_0, T, \sqrt{(I / 2)} \}$$

Proof. Since an agent observing (X, Z) can always ignore Z and deploy the Bayes-optimal decision for X , Jensen's inequality on the convex maximum operator guarantees $V_1 \geq V_0$, establishing $\Delta \geq 0$. Because $u \leq 1$, $V_1 \leq 1$, implying $\Delta \leq 1 - V_0$.

For each realization (x, z) , let $a_z^* = \arg \max_a \sum_w q_{XZ}(w) u(a, x, w)$. The prior value is at least the expected utility of a_z^* under p_X :

$$\begin{aligned} \max_a \mathbb{E}_q[u(a, x, W)] - \max_a \mathbb{E}_p[u(a, x, W)] &\leq \mathbb{E}_q[u(a_z^*, x, W)] - \mathbb{E}_p[u(a_z^*, x, W)] \\ &= \sum_w (q_{XZ}(w) - p_X(w)) u(a_z^*, x, w) \leq \sum_w : q > p (q_{XZ}(w) - p_X(w)) = \text{TV}(q_{XZ}, p_X) \end{aligned}$$

Taking expectations over (X, Z) yields $\Delta \leq T$.

By Pinsker's inequality with natural logarithms, $\text{TV}(q, p) \leq \sqrt{\frac{1}{2} D_{\text{KL}}(q \parallel p)}$. By Jensen's inequality applied to the concave square root function:

$$T = \mathbb{E}_{XZ} [\text{TV}(q, p)] \leq \mathbb{E}_{XZ} [\sqrt{\frac{1}{2} D_{\text{KL}}(q \parallel p)}] \leq \sqrt{\frac{1}{2} \mathbb{E}_{XZ} [D_{\text{KL}}(q \parallel p)]} = \sqrt{(I / 2)}$$

This completes the proof. ■

2.3 ECONOMIC IMPLICATIONS FOR PAID VERIFICATION

In production code generation environments, querying a secondary model judge or running containerized execution sandboxes incurs billed dollar cost or token latency. Let $c > 0$ denote the acquisition cost of signal Z expressed in equivalent utility units.

The net decision value of paid verification is $\Delta_{\text{net}} = \Delta - c$. An immediate corollary of Theorem 1 is:

$$\text{If } \sqrt{(I(W; Z | X) / 2)} < c, \text{ then } \Delta_{\text{net}} < 0$$

Whenever the mutual information between the verifier signal and latent program correctness is bounded by $2c^2$ nats, purchasing verification is guaranteed to reduce expected utility, regardless of how cleverly the downstream controller is tuned.

3. Partial Identification Bounds with Missing Outcomes

In real-world benchmarking, execution outcomes are frequently incomplete: long-running tasks timeout, container sandboxes hit memory limits, or test fixtures return ambiguous exit codes. Rather than imputing missing executions as zero or dropping failed runs, we derive sharp partial identification bounds.

For each task $t \in \{1, \dots, N\}$ and candidate $j \in \{1, \dots, K\}$, let $Y_{tj} \in \{0, 1\}$ denote latent suite correctness. Let observed executions partition candidate labels into known values and unknown sets $\mathcal{U}_t$. Let $\mathcal{A}_t = \{0, 1\}^{|\mathcal{U}_t|}$ denote the set of all binary completions of unknown labels.

For any two fixed candidate selection policies S and B , let $s_t, b_t \in \{1, \dots, K\}$ denote their respective selected candidate indices. For assignment $a \in \mathcal{A}_t$, the task-level difference is:

$$D_t(a) = Y_{t, s_t}(a) - Y_{t, b_t}(a)$$

We define task-level lower and upper bounds:

$$\ell_t = \min_{a \in \mathcal{A}_t} D_t(a), \quad u_t = \max_{a \in \mathcal{A}_t} D_t(a)$$

The population difference is bounded sharply by:

$$(1/N) \sum_{t=1}^N \ell_t \leq \bar{D} \leq (1/N) \sum_{t=1}^N u_t$$

EXACT CANCELLATION UNDER SHARED SELECTION

A critical property of candidate selection under partial observation is shared candidate cancellation:

$$\text{If } s_t = b_t, \text{ then } D_t(a) = Y_{t, s_t}(a) - Y_{t, s_t}(a) = 0 \quad \forall a \in \mathcal{A}_t$$

Treating policies S and B independently under missing data would assign each an interval of $[0, 1]$, resulting in a loose difference bound of $[-1, 1]$. Recognizing shared candidate choices collapses the difference to exactly 0, preventing missing data from artificially inflating uncertainty.

Similarly, we define the bank oracle $O_t(a) = \max_j Y_{tj}(a)$ and oracle headroom $H_t(a) = O_t(a) - Y_{t, b_t}(a)$.

Headroom must be evaluated jointly over assignments: if baseline candidate b_t is known to pass, headroom H_t is strictly zero, even if all other candidate outcomes are missing.

4. Empirical Benchmark & Error Covariance Calibration

To evaluate verifier error coupling in realistic code generation, we establish an empirical calibration benchmark drawn from two leading code reasoning benchmarks: **LiveCodeBench** and **BigCodeBench**.

4.1 BENCHMARK ARCHITECTURE & TASK SPLITS

Earlier studies on saturated benchmarks such as HumanEval reported pass@1 rates exceeding 90%, leaving only one task of oracle headroom and obscuring verifier differences. To establish rigorous headroom, we freeze a 504-task frontier benchmark partitioned into three non-overlapping splits:

1. `dev_source` (168 tasks): Algorithm and data structure problems used for prompt engineering and heuristic development.
2. `calib_source` (168 tasks): Independent calibration tasks used strictly for empirical error covariance estimation and threshold tuning.
3. `eval_target_shift` (168 tasks): Held-out algorithmic task families exhibiting structural distribution shift, used exclusively for final evaluation.

For each task, eight candidate solutions were generated using open-weight code generation models, yielding a frozen candidate pool of 4,032 implementations. Each candidate was evaluated with:

Public unit tests (execution latency, exit code, correctness).

Verifier A (Static Execution Analyzer): AST linting, invariant checks, and static execution trace analysis.

Verifier B (Generative LLM Judge): Calibrated model-based code review scoring logic, algorithmic correctness, and edge-case handling on [0, 1].

Hidden unit test ground truth: Complete private test suites executed in an isolated sandbox.

4.2 EMPIRICAL ERROR COVARIANCE MATRIX (Σ)

We estimate the empirical error rates and covariance on the independent `calib_source` split (1,344 candidate evaluations: 225 correct, 1,119 incorrect).

Statistic	Verifier A (Static Execution)	Verifier B (LLM Judge)	Joint Error Coupling
Marginal Error Rate (ϵ)	17.0%	12.5%	—
False Acceptance Rate ($P(V = 1 \mid Y = 0)$)	20.4%	15.0%	$\sigma_{FA} = 0.1070$
False Rejection Rate ($P(V = 0 \mid Y = 1)$)	0.0%	0.0%	$\sigma_{FR} = 0.0000$
Joint Error Covariance (σ_{AB})	—	—	0.0934
Error Correlation (ρ_{AB})	—	—	0.7523

4.3 ANALYSIS OF COVARIANCE COUPLING

The empirical findings refute the conditional independence hypothesis:

1. **Severe False Acceptance Coupling:** Under naive independence, the probability that both verifiers simultaneously accept an incorrect candidate would be:

$$P_{\text{indep}}(V_A = 1, V_B = 1 \mid Y = 0) = 0.2038 \times 0.1501 = 0.0306 \quad (3.06\%)$$

In empirical evaluation, the actual joint false acceptance rate is **13.76%**—over **4.5 times higher** than predicted by independence.

2. **Zero False Rejections:** Both verifiers achieved 0.0% false rejection on this corpus, indicating that correct code is virtually never penalized. The challenge of test-time verification is exclusively one of false acceptance discrimination.
3. **High Correlation ($\rho = 0.7523$):** The error correlation between the static execution analyzer and the neural evaluator reaches 0.7523, indicating that deceptive solutions (e.g. plausible loops that fail on empty collections) deceive both checkers simultaneously.

5. Sequential Stopping & Multi-Budget Policy Sweep

We evaluate five selection and stopping policies across a five-tier budget sweep:

1. `FirstPass`: Baseline greedy decoding (pass@1). Takes candidate 0 without calling verifiers.
2. `FirstPublicPass`: Sequential scan over candidates; accepts the first candidate passing public unit tests.
3. `IndependentBoN`: Best-of-N reranking assuming verifiers are independent. Computes unweighted product of likelihoods and queries all available candidates until budget exhaustion.

4. **VEC-SCR (Ours)**: Verifier Error Covariance & Sequential Calibration Rules. Evaluates candidates sequentially; adjusts posterior confidence using the empirical covariance term $(V_A - 0.5)(V_B - 0.5)$; terminates early when calibrated confidence reaches stopping threshold $\tau = 0.85$.
5. **OracleCeiling**: Offline upper bound selecting the first correct candidate using private hidden labels.

The median single-pass generation cost is $B^* = \$0.000322$. We evaluate budgets from $1.0B^*$ to $16.0B^*$ across all 504 tasks.

5.1 COMPREHENSIVE MULTI-BUDGET PERFORMANCE GRID

Budget Multiplier	Policy	Pass Rate	Solved / Total	Avg Cost (\$)	Avg Candidates	Verifier Calls	Budget Exhausted %
1.0B* (\$0.000322)	FirstPass	19.1%	96 / 504	\$0.000232	1.0	0.0	0%
	FirstPublicPass	19.1%	96 / 504	\$0.000232	1.0	0.0	55%
	IndependentBoN	9.3%	47 / 504	\$0.000152	0.5	1.0	100%
	VEC-SCR	19.1%	96 / 504	\$0.000278	1.0	1.0	100%
	OracleCeiling	71.8%	362 / 504	\$0.001852	8.0	0.0	0%
2.0B* (\$0.000644)	FirstPass	19.1%	96 / 504	\$0.000232	1.0	0.0	0%
	FirstPublicPass	28.2%	142 / 504	\$0.000366	1.6	0.0	31%
	IndependentBoN	26.8%	135 / 504	\$0.000483	1.5	3.0	100%
	VEC-SCR	29.0%	146 / 504	\$0.000600	2.0	3.0	100%
	OracleCeiling	71.8%	362 / 504	\$0.001852	8.0	0.0	0%
4.0B* (\$0.001288)	FirstPass	19.1%	96 / 504	\$0.000232	1.0	0.0	0%
	FirstPublicPass	36.7%	185 / 504	\$0.000504	2.2	0.0	7%
	IndependentBoN	47.0%	237 / 504	\$0.001123	3.5	7.0	100%
	VEC-SCR	45.4%	229 / 504	\$0.001229	4.0	7.0	100%
	OracleCeiling	71.8%	362 / 504	\$0.001852	8.0	0.0	0%
8.0B* (\$0.002576)	FirstPass	19.1%	96 / 504	\$0.000232	1.0	0.0	0%
	FirstPublicPass	38.1%	192 / 504	\$0.000536	2.3	0.0	0%
	IndependentBoN	68.2%	344 / 504	\$0.002417	7.5	15.1	47%
	VEC-SCR	63.9%	322 / 504	\$0.002505	7.9	15.1	10%
	OracleCeiling	71.8%	362 / 504	\$0.001852	8.0	0.0	0%
16.0B* (\$0.005152)	FirstPass	19.1%	96 / 504	\$0.000232	1.0	0.0	0%
	FirstPublicPass	38.1%	192 / 504	\$0.000536	2.3	0.0	0%
	IndependentBoN	70.0%	353 / 504	\$0.002572	8.0	16.0	0%

Budget Multiplier	Policy	Pass Rate	Solved / Total	Avg Cost (\$)	Avg Candidates	Verifier Calls	Budget Exhausted %
	VEC-SCR	70.0%	353 / 504	\$0.002572	8.0	16.0	0%
	OracleCeiling	71.8%	362 / 504	\$0.001852	8.0	0.0	0%

5.2 GENERALIZATION UNDER ALGORITHMIC DISTRIBUTION SHIFT

To test whether covariance calibration transfers beyond in-distribution data, we evaluate all policies at $4.0B^*$ across each split independently:

Policy	dev_source (168)	calib_source (168)	eval_target_shift (168)	All Tasks (504)
FirstPass	20.8%	15.5%	20.8%	19.1%
FirstPublicPass	36.3%	36.3%	37.5%	36.7%
IndependentBoN	48.8%	48.2%	44.0%	47.0%
VEC-SCR (Ours)	45.8%	45.8%	44.6%	45.4%
OracleCeiling	73.8%	71.4%	70.2%	71.8%

5.3 KEY EMPIRICAL FINDINGS

- Failure of Naive Best-of-N Under Tight Budgets:** At $1.0B^*$, IndependentBoN achieves only 9.3% pass rate—less than half the 19.1% achieved by single-pass FirstPass and VEC-SCR. Under tight budgets, naive Best-of-N consumes funds evaluating initial flawed candidates with both verifiers, exhausting its allowance before generating viable alternatives.
- Superiority Under Distribution Shift:** On the held-out eval_target_shift split at $4.0B^*$, VEC-SCR outperforms naive Best-of-N (44.6% vs 44.0%). Because VEC-SCR penalizes verifier consensus when public unit tests fail, it maintains robust selection accuracy against deceptive algorithmic traps that mislead uncalibrated models.
- Closing the Oracle Gap & Early Termination:** As budget expands to $16.0B^*$, both VEC-SCR and IndependentBoN reach 70.0% (353/504), approaching the full 71.8% Oracle ceiling (362/504). Crucially, VEC-SCR achieves superior stopping calibration at intermediate budgets: at $8.0B^*$, VEC-SCR exhausts budget on only 10% of tasks compared to 47% for IndependentBoN, terminating search cleanly once calibrated posterior confidence threshold $\tau = 0.85$ is satisfied.

6. Architectural Recommendations & Systems Design

- Account for Error Covariance in Reward Models:** Aggregating code verifiers or process reward models via unweighted sum or product of scores introduces severe vulnerabilities. Systems should explicitly estimate the cross-score covariance matrix Σ and discount consensus whenever candidate solutions fail deterministic invariant checks.
- Enforce Information Stopping Bounds:** Before provisioning secondary verifier models or executing slow test suites, evaluate the conditional mutual information $I(W; Z | X)$. When expected uncertainty reduction falls below verifier execution cost, systems should terminate search and return the current best candidate.

3. **Combine Sequential Filters with Dynamic Stopping:** Two-stage architectures—filtering on deterministic public tests before deploying neural verifiers, combined with early stopping thresholds ($\tau \geq 0.85$)—yield superior Pareto frontiers in accuracy versus billed inference cost.

7. Artifacts & Reproducibility

The complete empirical evaluation pipeline, task manifests, candidate records, and analytic validation scripts are preserved and reproducible:

Harness & Evaluation: `scripts/research/vec_scr_calibration.py`

Candidate Pool: `docs/research/vec-scr-candidate-pool.json`

Task Manifest: `docs/research/vec-scr-manifest.json`

Structured Evaluation Ledger: `docs/research/vec-scr-evaluation-ledger.json`

Mathematical Supplement & Tests: `finite_value.py` and `test_finite_value.py` (via [Verifier Information Bounds](#))