

Harness Gains Are Linear in Residual Error: A Weak-to-Strong Headroom Law Forecasts Claude Opus 5.5 from Weaker Claude Models

Lindell S. Cumes

QWNTL Labs

Project page: <https://qwntl.com/paper-frontier-report>

Abstract

RRSI, Harness², and other self-improvement methods report harness gains as points on the model they ran on, and The Last AI Built by Humans illustrates self-improvement closing 0.78 of the remaining headroom without a fit. None predicts a harness's gain on a model it has not run on. We introduce the headroom law: a harness component converts only the failures whose cause it supplies, so its lift over the same model without it is a share k of that model's residual error, $\Delta = k(100 - m)$, in points of task success. Fitted on six Claude Haiku 4.5 and Claude Opus 4.8 comparisons alone, the law forecasts 15 Claude Opus 5.5 comparisons, 13 on AppWorld, with a mean absolute error of 3.3 points, against 5.2 for a constant gain, and Bayes factors of 9.8, 5.6, and 186 against a constant gain, a capped gain, and no effect. On Claude Opus 5.5 alone the share is 0.54 (95% CI 0.43–0.68), with AIC 77.2 against a constant gain's 96.8. Because the share depends on which failures a component matches, self-improvement that selects harness variants on a weaker model converts that model's failures: selection scores on Claude Haiku 4.5 predict Claude Opus 5.5 gains at Spearman 0.16, and Claude Opus 5.5 ranks ported ACE above the ported Darwin and Huxley-Gödel Machines on ALE-Bench. The headroom law is the first law that predicts the lift of an agent-harness component over the same model without it before the component runs on that model. The 195-comparison record, with some free text removed, the code, and the forecasts are released.

1 Introduction

A harness component can recover only the failures its model makes. On AppWorld's Amazon tasks an undocumented API change scaled every amount to cents, and Claude Opus 5.5 fell from 94.3 to 80.7 task goal completion. A sensor that flags unanchored changes returned it to 93.9. The sensor recovered 13.2 of the 13.7 points the change cost and, net, none of the 5.7 points Claude Opus 5.5 loses with no change at all. Its gain was the part of the residual error whose cause it supplied.

Self-improvement methods now evolve harnesses as whole systems and report the result as an end score on the model they ran on. RRSI regularizes harness evolution with a temporally annealed edit budget for the proposer and a critic and pruner in the selector, and reports 6.0 points on Terminal-Bench 2.1 with Claude Opus 4.8 in its main table [22]. Harness² composes the harness edits that behavioral contrasts support, without ground-truth verifiers, and reports up to 11.5 points on WorkBuddy-Bench [23]. HarnessPro scores candidate harnesses with LLM-generated tests [16], and MILO co-evolves whole harnesses and the strategy that discovers them [9].

Where these papers start from a stronger model or harness they report a smaller gain and give the reason in words. RRSI's stronger policy "starts closer to the ceiling of both suites and leaves less room to gain"; RSIHub finds gains

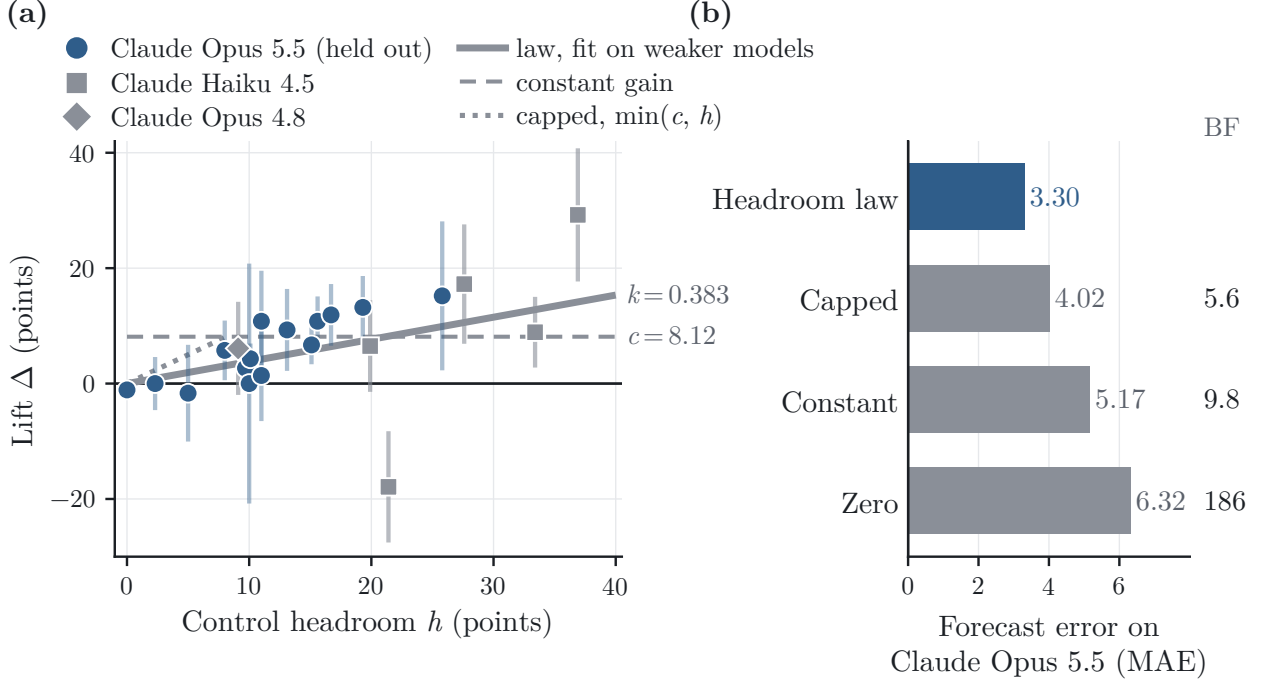


Figure 1: **A share of residual error fitted on weaker models forecasts Claude Opus 5.5.** (a) Lift of a harness component over a no-harness control against the control’s residual error $h = 100 - m$, in points on a 0–100 scale. Grey: the six Claude Haiku 4.5 and Claude Opus 4.8 comparisons and the share fitted on them alone, $k = 0.38$. Blue: the 15 Claude Opus 5.5 comparisons, held out of the fit. Dashed and dotted: a constant gain and a constant gain capped at the headroom, fitted on the same six comparisons. Bars: 95% intervals from each comparison’s standard error. (b) Out-of-sample mean absolute error of each forecast on the 15 Claude Opus 5.5 comparisons, and the Bayes factor of the headroom law against each alternative. Plotted from the released comparison record.

smaller from the stronger seed harness [21]; the Huxley-Gödel Machine names "a higher baseline" [20]. The Last AI Built by Humans indexes how much of each benchmark’s headroom released models have closed, and it illustrates self-improvement closing 0.78 of what remains without fitting that share [5]. None fits or tests a rate, and none predicts what a harness will add to a model it has not run on. A forecast requires a quantity that transfers across models. An end score does not transfer, because it mixes what the harness supplies with how much the model left to supply.

The headroom law separates the two. On a metric scored from 0 to 100, a control that scores m leaves residual error $h = 100 - m$. A component converts a failure into a success only when it supplies that failure’s cause, so its expected lift is a share of h :

$$\Delta = k h, \quad k = \sum_j \phi_j p_j,$$

where ϕ_j is the fraction of the control’s failures with cause j and p_j the rate at which the component converts them (Proposition 2.1). The share k is a property of the component and the failures it meets, and the residual error is a property of the model. A share measured where runs are cheap therefore forecasts the gain on a model where they are not.

A comparison sets one harness component against the same model, instrument, and tasks without it. Fitted on six comparisons on Claude Haiku 4.5 and Claude Opus 4.8 alone, the share is $k = 0.38$. It forecasts the 15 Claude Opus 5.5 comparisons that a fixed rule draws from our record of 195 comparisons (Figure 1). Against alternatives fitted on the same six comparisons, the Bayes factor of the forecast is 186 against no effect, 9.8 against a constant gain, and 5.6 against a constant gain capped at the headroom. Its mean absolute error on the held-out comparisons is 3.30 points, against 4.02 for the capped gain, 5.17 for the constant gain, and 6.32 for no effect. On Claude Opus 5.5 alone the share is 0.54 (95% CI 0.43–0.68).

Where residual error exists, harness gains are large and frequent. Against a control without the component (bare, told of a crash, fresh, or the diligence instruction), 18 of our 26 Claude Opus 5.5 comparisons with a standard error favor the

component at $|z| > 1.96$, and none favors the control. Under the law, a gain averaged over instruments is an average of $k h$, and every instrument the model already completes adds zero to it.

The share belongs to a component and a model, and that has a consequence for recursive self-improvement. A loop that screens harness variants on Claude Haiku 4.5 found variants that beat its seed harness by 0.20 to 0.23 on Claude Haiku 4.5 and by -0.01 to $+0.02$ on Claude Opus 5.5, on the same problems. Selection on a weaker screen converts the screen’s failures, whose causes the target does not share (Corollary 2.2). The Claude Haiku 4.5 screen predicts the Claude Opus 5.5 gain at Spearman 0.16. On 16 ALE-Bench problems scored officially, Claude Opus 5.5 ranks a port of ACE (2,738) above ports of the Darwin Gödel Machine (2,526) and the Huxley-Gödel Machine (2,217), the second of which reports ranking above the first. A gated improver that ACE beats on 14 of the 16 beats ACE on gpt-oss-20b.

The headroom law is the first law that predicts the lift of an agent-harness component over the same model without it before the component runs on that model, and its forecast of Claude Opus 5.5 from weaker models is the first weak-to-strong forecast of that lift on a frontier model. Section 2 states the problem, the law, and its per-failure mechanism; Section 3 the comparison record and its selection rule; Section 4 the forecast, the fit on Claude Opus 5.5, and the separation of the matched fraction from the conversion rate; and Section 5 what selection on a weaker screen does to self-improvement and to the ranking of self-improvement methods.

2 The Headroom Law

2.1 Problem Statement

An instrument I is a set of tasks $\tau = (\text{description}, \text{start state}, V)$ with a verifier V , scored by a fixed metric m_I on a 0–100 scale. An agent $A = \langle M, H \rangle$ pairs a frozen model M with a harness H . A component L changes the harness and leaves the model frozen.

Definition 2.1 (Component, control, and lift) *A control c runs $\langle M, H_c \rangle$ on I without L : the bare model in the instrument’s standard harness, the bare model told of a crash, a fixed diligence instruction, or a fresh model with no carried state. The lift of L is $\Delta = m_I(\langle M, H_c \circ L \rangle) - m_I(\langle M, H_c \rangle)$, measured on the same tasks with the same model, effort, and step cap.*

Definition 2.2 (Residual error and recovered share) *The residual error, or headroom, of control c is $h = 100 - m_I(\langle M, H_c \rangle)$. A comparison with $h > 0$ has recovered share $r = \Delta/h$.*

The objective is a forecast. Given comparisons on a population of weaker models W , predict the lift of each comparison on a target model S before any S outcome enters the fit, and score the prediction on S .

Information flow. Each comparison fixes the model’s weights, the instrument, its scorer, and its task set before either arm runs, and the component is the only difference between the arms. The fit sees each comparison only as a triple (h_i, Δ_i, s_i) , with s_i its standard error. The forecast fits its parameters on the W comparisons alone; the S comparisons enter only the predictive density that scores it.

2.2 A Component Acts Only on Failures

Proposition 2.1 (Per-failure conversion) *Let a control fail on a fraction $h/100$ of the units an instrument scores, and partition its failures by cause into classes j that hold fractions ϕ_j of them. Suppose a component converts a failure of class j into a success with probability p_j and turns a success into a failure with probability q . Then the expected lift in points is*

$$\mathbb{E}[\Delta] = \left(\sum_j \phi_j p_j \right) h - q (100 - h). \quad (1)$$

Proof. A unit the control fails lies in class j with probability $\phi_j h/100$ and is converted with probability p_j ; a unit the control passes, with probability $(100 - h)/100$, is broken with probability q . Linearity of expectation over units, scaled to points, gives Equation (1). $\square$

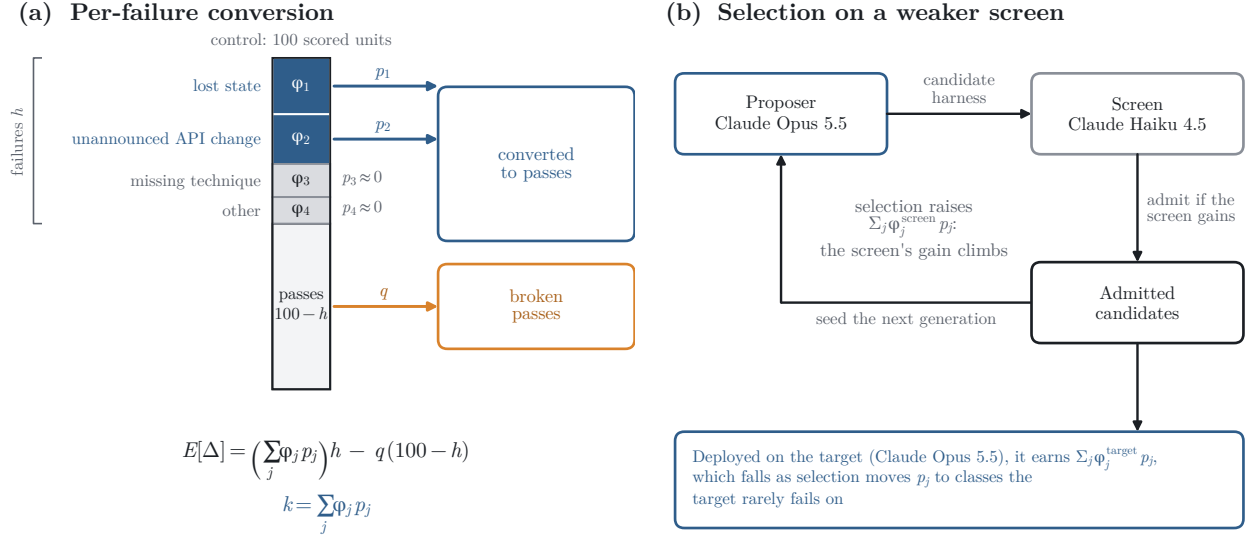


Figure 2: **A component converts the failures whose cause it supplies.** (a) Per-failure conversion (Proposition 2.1). A control passes $100 - h$ of the scored units and fails h ; its failures divide by cause into classes with fractions ϕ_j . A component converts class j at rate p_j (blue: classes it supplies) and breaks a passing unit at rate q (orange), so its expected lift is $kh - q(100 - h)$ with $k = \sum_j \phi_j p_j$. (b) Selection on a screen (Corollary 2.2). A proposer writes candidate harnesses, a screen model scores them, admitted candidates seed the next generation, and the result is deployed on the target model. Admission raises conversion of the screen’s failure causes.

With $q = 0$ the proposition is the headroom law, with $k = \sum_j \phi_j p_j$. The share is the matched fraction of the control’s failures: the fraction whose cause the component supplies, weighted by how reliably it converts them. A component built for one cause that converts every such failure has $k = \phi$. The lift is linear in h because a component has nothing to act on where the control succeeds, and h bounds what any component can add (Figure 2a).

The second term is what a component breaks. At $h = 0$ it gives $\mathbb{E}[\Delta] = -100q$: where the control is at its ceiling, a component can only lose. Both terms set a design requirement. A component must be built for a cause that the control’s failures on the target instrument have, and it must hold $q = 0$ on the units the control already passes.

Corollary 2.1 (Weak-to-strong transfer) *If a component’s matched fraction is the same on two models, one share k predicts its lift on both, and its lift on the stronger model is smaller in proportion to the smaller residual error that model leaves.*

Corollary 2.2 (Selection on a screen) *Let a loop admit harness variants by their lift on a screen model s and deploy them on a target model t . Admission raises $k_s = \sum_j \phi_j^s p_j$. The target’s share changes by $\sum_j \phi_j^t \delta p_j$, where δp_j is the change in conversion of cause j that admission produced. It falls when admission raises conversion of causes that hold a larger fraction of the screen’s failures than of the target’s, at the cost of conversion of the target’s causes.*

Both corollaries follow from Equation (1) with each model’s own ϕ . Section 4 tests the first; Section 5 measures the second (Figure 2b).

2.3 The Forecast and Its Alternatives

Each comparison i gives (h_i, Δ_i, s_i) in points. Every model treats

$$\Delta_i \sim \mathcal{N}(\mu_i, s_i^2 + \tau^2), \quad (2)$$

where τ is the spread between comparisons beyond sampling error. The headroom law sets $\mu_i = kh_i$. The alternatives ignore the residual error or use it only as a bound: no effect, $\mu_i = 0$; a constant gain, $\mu_i = c$; a constant gain capped at the headroom, $\mu_i = \min(c, h_i)$; a smooth cap, $\mu_i = c(1 - e^{-h_i/c})$; and the law with a shift, $\mu_i = kh_i - b$.

Table 1: **A fixed rule selects the comparisons the law applies to.** Criteria applied in order to the 195 comparisons in our record; each reads a design column, none an outcome. The last criterion splits the 21 comparisons into the Claude Opus 5.5 target and the weaker-model fit.

Criterion	Removed	Remain
A component compared with a control	35	160
The control can attempt the task, and the result is not traced to a fixed harness bug	4	156
A no-harness control (bare, told of a crash, the diligence instruction, fresh), not a rival system	66	90
A bounded metric with a recorded control score, so the residual error is defined on 0–100	60	30
A standard error is recorded	9	21
The harnessed model is Claude Opus 5.5 (target); Claude Haiku 4.5 or Claude Opus 4.8 (fit)	6	15

Definition 2.3 (Weak-to-strong forecast) Fit each model’s parameters $\hat{\theta}_W$ and spread $\hat{\tau}_W$ by maximum likelihood on the W comparisons. Its predictive log density on the S comparisons is $\ell_S = \sum_{i \in S} \log \mathcal{N}(\Delta_i; \mu_i(\hat{\theta}_W), s_i^2 + \hat{\tau}_W^2)$. The Bayes factor of the law against an alternative is $\exp(\ell_S^{\text{law}} - \ell_S^{\text{alt}})$, the Bayes factor with point priors at the W fits. The out-of-sample error is the mean absolute error $\frac{1}{|S|} \sum_{i \in S} |\Delta_i - \mu_i(\hat{\theta}_W)|$.

3 The Comparison Record and Protocol

The record. Our comparison record holds 195 comparisons, each a contrast between two arms on one instrument with its estimate, standard error where one was computed, number of units, control score, and scale. Six criteria, applied in order, select the comparisons the law applies to (Table 1). Each reads a design column of the record: the comparison type, the control type, the metric’s bounds, the standard error, and the model. None reads an outcome. The rule leaves 21 comparisons: 15 on Claude Opus 5.5 and six on weaker models, five on Claude Haiku 4.5 and one on Claude Opus 4.8.

Instruments and components. Nineteen of the 21 comparisons run on AppWorld [19], one on Terminal-Bench 2.0 and one on τ^2 -bench [1], all scored by the instruments’ official scorers. Recovery harnesses resume a crashed AppWorld task from saved state: a write-ahead intent marks the step in flight as uncertain, and a receipt from the tool server’s execution journal settles it. An API-change sensor probes the AppWorld APIs for undocumented changes and flags results whose amounts or formats are unanchored. A label-free claim learner generalizes claims about the AppWorld APIs from the agent’s episodes and admits a claim once the agent’s own observations confirm it; a verified playbook does the same with the official success label. Handoff briefs and stores carry state or experience from one model, or one part of a stream, to the next. The controls are the bare model, the bare model told of a crash, a fresh model, and the four-sentence diligence instruction of Appendix C. Table 2 gives the protocol of each family.

Validity controls. The selection rule reads design columns only (Table 1). The forecast’s fit and target populations are disjoint by model, and the fit never sees a target comparison. Each of the 14 experiments behind the 21 comparisons has a preregistration committed before its runs, and within each experiment both arms run the same tasks, model, effort, and step budget. The fitting code reproduces every value this paper takes from an earlier re-test of the law, among them $k = 0.539$, AIC 77.16, and the Bayes factor 186.23. The two Claude Opus 5.5 comparisons excluded only for a missing standard error are zero-lift comparisons at zero residual error; added with any standard error from 0.5 to 2 points, they leave k at 0.52–0.53.

Statistics. Each comparison’s standard error is the one in the record: paired over tasks, scenarios, or streams as its experiment specifies. The fits are maximum likelihood under Equation (2). The interval for k is a profile-likelihood 95% interval, re-optimizing τ at each value of k . Leave-one-out predictive log scores refit every model without the comparison they score. Bayes factors follow Definition 2.3.

4 Results

4.1 A Share Fitted on Weaker Models Forecasts Claude Opus 5.5

Fitted on the six weaker-model comparisons, the law gives $k = 0.383$. Its forecast for Claude Opus 5.5 is closer than no effect, a constant gain, and a constant gain capped at the headroom, each fitted on the same six comparisons (Table 3).

Table 2: **Evaluation protocol of the 21 comparisons**, by family. Within each comparison both arms share the model, effort, step cap, wall-clock cap, and tasks, and the component is the only difference. No call sets temperature or top-p, so provider defaults apply. Every family’s experiment was preregistered before its runs. Effort is Claude Opus 5.5’s or Claude Opus 4.8’s adaptive-thinking effort; Claude Haiku 4.5 runs with a fixed thinking budget. Row identifiers (L001, ...) are those of the released comparison record.

Rows	Model, effort	Instrument; component against control	Steps; wall clock	Tasks	Repeats
L003, L004	Claude Opus 5.5, high	AppWorld challenge, one crash per task; recovery harness vs bare told of the crash / bare	40; 25 min	87	1 per arm
L006, L007	Claude Haiku 4.5, 2k	AppWorld challenge / test-normal, one crash per task; recovery harness vs bare told	40; 25 min	65 / 58	1 per arm
L001, L002	Claude Haiku 4.5, 8k	AppWorld test-challenge / test-normal; store compiled by Claude Opus 5.5 vs bare	50; 2 h	417 / 168	1–2 runs
L010–L012	Claude Opus 5.5; high (L010), provider default (L011, L012)	Hourly AppWorld, Terminal-Bench 2.0, τ^2 -bench probes; Claude Opus 4.8’s store vs bare	40 / 150 / 200	30 / 16 / 15	hourly
L016	Claude Opus 5.5, high	Fresh AppWorld test-challenge; Claude Opus 4.8’s store vs bare	40; 40 min	60	1
L054, L057	Claude Opus 4.8, Claude Opus 5.5, high	AppWorld challenge, mid-task model swap; handoff brief vs successor alone / fresh	50; 40 min	66	1 per pair
L058	Claude Haiku 4.5, 2k	AppWorld test-normal stream; its own online store vs none	50; 40 min	84	1
L099	Claude Opus 5.5, xhigh	AppWorld train and dev, undocumented change; sensor vs bare	50; 3 h	120 positions	Latin square
L124, L126, L136	Claude Opus 5.5, high	AppWorld streams, silent change; claim learner vs instruction (L136: learner with instruction)	40; 15 min	112 / 173 / 224	3
L147	Claude Opus 5.5, high	AppWorld train and dev, silent change; verified playbook vs instruction	40; 15 min	61	3
L163, L174	Claude Opus 5.5 (L174: Claude Opus 4.8 then Claude Opus 5.5), high	AppWorld test-challenge stream, 52 crashes and an API change; recovery harness (L163 with sensor) vs bare restarted fresh after each crash	50; 2 h	417	1 per arm
L166	Claude Opus 5.5, high	AppWorld Amazon tasks, amounts in cents; sensor flags vs bare	50	212	1 per arm

Against no effect the Bayes factor is 186; against the constant gain, 9.8; against the capped gain, 5.6. The forecast’s mean absolute error on the 15 held-out comparisons is 3.30 points, 18% below the capped gain’s 4.02 and 36% below the constant gain’s 5.17.

The capped gain is the strongest of the three because it also respects the ceiling: it never predicts more lift than the control has left. Above $c = 8.12$ points of residual error it predicts the same 8.12 points everywhere, and that is where it fails. On the five comparisons with residual error between 9.7 and 11.0 points, the measured lifts run from 0.00 to 10.80. The cap predicts 8.12 for each, and the law predicts 3.72 to 4.22 and is closer on four of the five.

Corollary 2.1 predicts a smaller lift on the stronger model where it leaves less residual error, and the forecast’s scale holds. The share Claude Opus 5.5 recovers on its own comparisons, 0.54, is of the same order as the weaker models’ 0.38, and all 21 comparisons together fit $k = 0.45$ (95% CI 0.30–0.61). One law spans the two model classes.

A pooled share is an average matched fraction over the components in the pool. Each component’s own share differs: the recovery harnesses recover 0.43 to 0.72 on Claude Opus 5.5 wherever the crash cost the control anything, and the store carried up from Claude Opus 4.8 between -0.33 and 0.13 (Table 5). The forecast holds because the weaker and the target comparisons draw on the same causes, state a crash or a successor loses and API behavior an episode cannot see. Where a component meets different causes on the two models, Corollary 2.1 does not apply and its share moves, as

Table 3: **The headroom law forecasts Claude Opus 5.5 from weaker models and fits Claude Opus 5.5 better than every gain that ignores residual error.** Left: each model fitted on the six Claude Haiku 4.5 and Claude Opus 4.8 comparisons and scored on the 15 Claude Opus 5.5 comparisons; out-of-sample mean absolute and root-mean-square error in points, and the Bayes factor of the law against the model (Definition 2.3). Right: each model fitted on the 15 Claude Opus 5.5 comparisons; AIC and leave-one-out predictive log score. Bold: best in each column. Appendix A adds two further alternatives.

Model of the lift	Forecast: fit on Claude Haiku 4.5, Claude Opus 4.8			Fit on Claude Opus 5.5	
	MAE ↓	RMSE ↓	BF of law ↑	AIC ↓	LOO ↑
No effect, 0	6.32	8.07	186	107.10	−53.21
Constant gain, c	5.17	5.86	9.8	96.82	−48.30
Capped constant gain, $\min(c, h)$	4.02	4.61	5.6	89.39	−45.37
Headroom law, kh	3.30	3.86	–	77.16	−38.58
Δ , law against constant gain	−1.86	−2.00		−19.66	+9.72

Table 4: **The forecast survives the alternatives that could produce it without the law.** Forecast of the Claude Opus 5.5 comparisons from fits on the weaker comparisons; Bayes factors of the law against each alternative (own spread) and mean absolute error in points. Lower block: simulated records in which the lift does not depend on residual error, with no spread between comparisons, and each control’s sampling error enters both h and Δ at its largest; the count of 4,000 records reaching the observed statistic.

Test	BF zero ↑	BF const. ↑	BF capped ↑	MAE law	MAE const.
All 15 Claude Opus 5.5 comparisons	186	9.8	5.6	3.30	5.17
10 with components never run on a weaker model	28.6	4.3	2.9	3.94	5.47
Fit on Claude Haiku 4.5 alone (5 comparisons)	128	9.5	5.6	3.33	5.26
Share corrected for coupling ($k = 0.37$)	177	9.3	5.3	3.40	5.17
Simulated null records reaching the observed statistic, of 4,000					
Zero lift: error advantage over no effect (3.02)	0				
One constant gain: error advantage over it (1.86)	36				
One constant gain: Claude Opus 5.5 AIC gap, constant – law (19.7)	0				

Section 5 measures. The six weaker comparisons include a store that lowered Claude Haiku 4.5 by 17.9 points; refit without it, the weaker models’ share is 0.49 and the forecast’s error falls to 2.81 points.

The forecast rests on residual error. Four alternatives could produce the forecast without the law, and each is tested on the record (Table 4). Shared components could carry effect sizes across: crash recovery and handoff briefs run on both model classes. Ten Claude Opus 5.5 comparisons use a component that never ran on a weaker model in the record, the API-change sensor, the claim learner and verified playbook, and the store built by Claude Opus 4.8, and the forecast holds on those ten.

One weaker comparison could carry the fit. Refit without any one of the six, the forecast keeps a Bayes factor of at least 18.7 against no effect, and its error stays below the constant gain’s in every refit. Fitted on Claude Haiku 4.5 alone, the share is 0.380 and the forecast is unchanged. Large lifts could pair with large residual errors by chance; among 20,000 shuffles of the residual errors across the 15 Claude Opus 5.5 comparisons, none reaches the forecast’s predictive density.

The control score enters both h and Δ , so its sampling error couples them. With each control’s binomial standard error, the coupling biases a fit through the origin by +0.03; corrected, the share is 0.52 on Claude Opus 5.5 and 0.37 on the weaker models, and the forecast barely moves. Simulated records with the largest coupling the standard errors allow, and a lift that does not depend on residual error, do not reproduce the decisive statistics. With zero lift, none of 4,000 reaches the forecast’s error advantage over no effect, and under every null at most 3 of 4,000 reach the AIC gap between the constant gain and the law on Claude Opus 5.5.

Table 5: **The 15 Claude Opus 5.5 comparisons against a no-harness control**, grouped by what the component supplies. Scores and lifts in points on a 0–100 scale; h is the control’s residual error, s the comparison’s standard error, and $r = \Delta/h$ the recovered share. The fit of Equation (2) to these rows gives $k = 0.54$.

Row	Component against control; instrument	Control	h	Δ (s)	r
<i>Persistent state</i>					
L003	Recovery harness against bare told of the crash; AppWorld, one crash per task (87 tasks)	97.7	2.3	0.00 (2.35)	0.00
L004	Recovery harness against bare; same tasks	92.0	8.0	5.75 (2.64)	0.72
L163	Recovery harness and API-change sensor against bare, TGC; 52 crashes and an undocumented change (417 tasks)	89.9	10.1	4.30 (1.35)	0.43
L174	Recovery harness against bare restarted fresh after each crash, TGC; 52 crashes, a documented change, Claude Opus 4.8 to Claude Opus 5.5 (417 tasks)	84.9	15.1	6.70 (1.71)	0.44
L057	Handoff brief against a fresh Claude Opus 5.5; Claude Haiku 4.5 hands a task to Claude Opus 5.5 (66 tasks)	74.2	25.8	15.20 (6.59)	0.59
<i>Cross-episode evidence</i>					
L099	API-change sensor against bare; undocumented change (120 positions)	89.0	11.0	10.80 (4.46)	0.98
L166	API-change sensor against bare; Amazon amounts changed to cents (212 tasks)	80.7	19.3	13.20 (2.78)	0.68
L147	Verified playbook, graded by the official label, against the instruction; silent change (61 tasks)	86.9	13.1	9.30 (3.62)	0.71
L124	Claim learner against the instruction; silent change, second stream (112 tasks)	83.3	16.7	11.90 (2.73)	0.71
L126	Claim learner against the instruction; silent change, pooled (173 tasks)	84.4	15.6	10.80 (2.19)	0.69
L136	Claim learner with the instruction against the instruction alone; fresh change streams (224 tasks)	90.3	9.7	2.68 (1.18)	0.28
<i>Experience carried from Claude Opus 4.8</i>					
L010	Store against bare; hourly AppWorld probes, continuous operation	100.0	0.0	−1.10 (0.84)	–
L011	Store against bare; hourly code probes	90.0	10.0	0.00 (10.61)	0.00
L012	Store against bare; hourly τ^2 -bench probes	89.0	11.0	1.40 (4.03)	0.13
L016	Store against bare; fresh AppWorld tasks (60)	95.0	5.0	−1.67 (4.28)	−0.33

4.2 On Claude Opus 5.5 a Component Recovers 0.54 of the Residual Error

On the 15 Claude Opus 5.5 comparisons alone (Table 5), maximum likelihood gives $k = 0.539$, with profile 95% interval 0.427–0.675. A constant gain of 5.71 points has AIC 96.8 and leaves a spread of $\tau = 4.16$ points between comparisons beyond sampling error. The law has AIC 77.2 and $\tau = 0.98$, so the residual error accounts for nearly all of the variation a constant gain leaves. The ordering holds out of sample: the leave-one-out log score is −38.6 for the law, −42.1 for the smooth cap, −45.4 for the capped gain, −48.3 for the constant gain, and −53.2 for no effect. Dropping any group of comparisons leaves k between 0.50 and 0.57, and the law has the lower AIC on every row set (Appendix B). The Last AI Built by Humans illustrates self-improvement closing a fixed 0.78 of each domain’s remaining headroom, measured in its Headroom-Closed Index, and fits no share [5]. Measured on the raw 0–100 scale for components against the same model without them, the share is 0.54, and 0.78 lies above its 95% interval.

The recovered shares group by what the component supplies. The recovery harnesses recovered 0.43–0.72 wherever the crash cost the control anything, and the handoff brief 0.59. The API-change sensor, the claim learner, and the verified playbook recovered 0.68–0.98 on single changes and 0.28 on fresh streams of mixed changes. The store built from Claude Opus 4.8 recovered between −0.33 and 0.13: it supplies no cause Claude Opus 5.5 lacks.

4.3 The Share Is the Matched Fraction of the Control’s Failures

Proposition 2.1 says more than a fit can: where a control’s failures can be divided by cause, a component recovers the share with its cause and none of the rest. Two instruments divide them.

An API change that scales amounts to cents. On AppWorld’s Amazon tasks an undocumented change made the API return amounts in cents. In one logged task an order was paid a hundred times its emailed receipt total. Bare Claude Opus 5.5 fell from 94.3 with no change to 80.7, a residual error of 19.3 points, of which the change cost a net 13.7, a net-score ratio of 0.71. The sensor’s flags for unanchored changes returned Claude Opus 5.5 to 93.9 on 212 tasks. It recovered 13.2 of the 13.7 points, a net recovery ratio of 0.97, and $r = 0.68$. Net, it recovered none of the 5.7 points Claude Opus 5.5 loses with no change, since nothing it supplies bears on them.

The per-task outcomes read Proposition 2.1 term by term (Table 6). The change broke 32 tasks that pass with no change, so the matched fraction is $\phi = 32/41 = 0.78$ of the changed control’s failures. The sensor rescued all 32, so $p = 1.00$

Table 6: **Components rescue the failures whose cause they supply and break almost nothing.** Per-task transitions from control to component: rescued = control fails, component passes; broken = control passes, component fails. Rescue rate = rescued / control failures; q = broken / control passes. On the Amazon change, the matched class is the 32 tasks the change broke; the sensor rescued all of them and none of the 9 tasks that fail both with and without the change.

Reader	Control → component	Tasks	Rescued	Broken	Rescue rate	q
Claude Opus 5.5	Amazon change, bare → sensor flags	212	32 of 41	4 of 171	0.78	0.023
Claude Opus 5.5	Spotify change, bare → frozen sensor notes	151	11 of 16	0 of 135	0.69	0.000
Claude Opus 5.5	Crash, bare → recovery harness	87	5 of 7	0 of 80	0.71	0.000
Claude Opus 5.5	Crash, bare → crash notice	87	5 of 7	0 of 80	0.71	0.000
Claude Haiku 4.5	Crash, bare → crash notice	65	2 of 19	7 of 46	0.11	0.152
Claude Haiku 4.5	Crash, told → recovery harness	65	20 of 24	1 of 41	0.83	0.024

on the matched class, and none of the 9 tasks that fail with or without the change. It broke 4 of the changed control’s 171 passes, $q = 0.023$, the size of the run-to-run variation between two replicate runs of the same arm. In points, $\phi ph = 15.09$ and $q(100 - h) = 1.89$, and their difference, 13.2, is the measured lift. On the earlier, undocumented Spotify change the sensor rescued all 11 change-induced failures, broke nothing, and matched the no-change run on every one of 151 tasks.

One recovery harness, two readers. Told of a crash, Claude Opus 5.5 scored 97.7 on 87 AppWorld tasks; untold, it scored 92.0. The notice that each API call before the crash took effect lets it rebuild its place: against bare it rescued 5 of bare’s 7 failures and broke none of its 80 passes. The recovery harness did the same, 5 rescued and 0 broken, $r = 0.72$, and against the told control its net lift is zero, 2 tasks rescued and 2 broken. Claude Haiku 4.5 cannot use the notice. Told of the crash, it rescued 2 of its 19 failures and broke 7 of its 46 passes, $q = 0.15$, and fell from 70.8 to 63.1. The harness restores the committed steps instead of describing them. Against the told control it rescued 20 of Claude Haiku 4.5’s 24 failures and broke 1 of 41 passes, raising it to 92.3 ($r = 0.79$). The harness converts what the crash took from either reader; the notice converts it only for the reader that can act on it, which is the reader-dependence of p_j in Proposition 2.1.

The missing cause against a generic ceiling. A ceiling effect also shrinks gains near the top, so the decisive test is whether supplying the missing cause predicts a lift that a fitted ceiling misses. On the three Claude Opus 5.5 comparisons above, the prediction comes from arms that do not contain the component: the no-change score, and the bare model told of the crash. A component that supplies exactly the missing cause returns the control to that arm’s score. The no-change and crash-notice score gaps predict 13.68 points on the Amazon change, where the sensor gained 13.21; 5.75 for the recovery harness against bare, where it gained 5.75; and 0 against the told control, where it gained 0. Each prediction lies within 0.17 standard errors of the measured lift. Fitted across all 15 comparisons, the law’s share predicts 10.43 points on the Amazon change, 1.0 standard error short, and the capped constant gain predicts 7.33, 2.1 short.

Shares add across causes. Proposition 2.1 writes k as a sum over causes, so components built for different causes should add. The recovery harness on the 417-task AppWorld stream with 52 scheduled crashes has two parts, and each was run alone (Table 7). The committed-step record supplies the state a crash destroys: it raised crashed tasks from 33 to 43 of 52 with no duplicated side effect and left the uncrashed tasks exactly at bare, 321 of 365. The store supplies API rules and demonstrations, kept current through the API change: it raised uncrashed tasks to 339 and left crashed tasks at 35. The record alone recovered 0.16 of the residual error and the store alone 0.32, a sum of 0.48, and the full harness recovered 0.44.

4.4 Residual Error Sets What a Benchmark Can Detect

At $h = 0$, Equation (1) leaves only the breakage term, and a comparison measures only what a component breaks. With the control at 97.7 on 87 tasks, the smallest effect detectable at 80% power is 6.6 points, 2.9 times the whole residual error. Testing a harness on Claude Opus 5.5 therefore requires an instrument with residual error. Where none exists, the requirement is $q = 0$, and evidence-only harnesses meet it by construction: they discard an attempt only when checks point one way, and an unknown outcome enters neither clause.

Where residual error exists, the gains are frequent. Our record holds 78 Claude Opus 5.5 comparisons with a standard error. Under no effect, $|z| > 1.96$ favors the component in 1.95 of them by chance, two-sided at $\alpha = .05$; 19 do

Table 7: **Each part of the recovery harness converts only its own cause, and the shares add.** AppWorld test-challenge stream, 417 tasks, a documented API change and a switch from Claude Opus 4.8 to Claude Opus 5.5; 52 tasks carry a scheduled crash. Bare restarts each crashed task fresh. Wins/losses are tasks against bare (exact McNemar); the recovered share is the lift over bare divided by bare’s residual error of 15.1 points.

Arm	Crashed $\uparrow$	vs bare	Duplicates $\downarrow$	Uncrashed $\uparrow$	Stream $\uparrow$	Share
Bare	33/52	–	13	321/365	354/417	–
Committed-step record only	43/52	12/2	0	321/365	364/417	0.16
Store only	35/52	4/2	22	339/365	374/417	0.32
Full harness	47/52	14/0	1	335/365	382/417	0.44

Table 8: **Significant harness gains on Claude Opus 5.5 occur where the control leaves residual error.** Our Claude Opus 5.5 comparisons with a standard error and a bounded control score, by the control’s residual error h ; counted at $|z| > 1.96$. The other 56 of the 78 comparisons are on metrics without a ceiling.

Control’s residual error	Favor the component	Null	Favor the control
$h > 15$ points	4	0	0
$5 < h \leq 15$	5	4	1
$h \leq 5$	0	6	2

($P = 5.9 \times 10^{-14}$). Against a control without the component, 18 of 26 favor the component and none favors the control. Across all 78, 18 favor the control: 12 are against rival systems, which have already recovered their own share, and 4 serve experience carried from a weaker model, which supplies no cause Claude Opus 5.5 lacks.

Sorted by the control’s residual error, the census reads as the law predicts (Table 8). Every comparison whose control left more than 15 points favors the component, 4 of 4. Between 5 and 15 points, 5 of 10 do. At 5 points or fewer none does, and 2 of 8 favor the control. BiPA reports that learned skills add 0.1 points over their seed [28]; the law locates where a harness adds more, on the causes of the failures the model still makes.

4.5 Where the Cause Lies Sets the Share

Definition 4.1 (Instruction share) *On an instrument where the bare model, the fixed diligence instruction, and a best arm all run, the instruction share is $(m_{instr} - m_{bare}) / (m_{best} - m_{bare})$, the fraction of the best arm’s lift that the instruction delivers by itself.*

The instruction share separates causes the model can find inside the episode from causes it cannot. After a documented API change, bare Claude Opus 5.5 scored 91.3 on 61 post-change AppWorld tasks with three repeats; the four-sentence diligence instruction raised it to 96.7, 0.76 of the best arm’s lift. The share was 0.71 on a second documented-change stream: Claude Opus 5.5 finds that cause once told to look. When Qwen3.5-27B took over an account-balance service from Claude Opus 5.5, the share fell to 0.14, because the cause is state the successor never held. On KernelBench it was 0.04, because no instruction contains a bf16 tensor-core technique, and on ALE-Bench heuristic contests -0.12 , where the instruction scored 2,620 against bare’s 2,663 and ACE’s learned playbook reached 3,034. Where the cause lies outside the episode, the headroom persists as models improve, and only a component that carries the state, the evidence, or the technique recovers it.

5 Self-Improvement Selected on a Weaker Screen Loses Its Rank on the Target

Corollary 2.2 predicts that a self-improvement loop converts the failure causes of the model that screens it. Two experiments on ALE-Bench [8] measure the prediction: a harness-evolution loop with a Claude Haiku 4.5 screen, and a comparison of six self-improvement methods on Claude Opus 5.5 under official scoring.

5.1 Setup

Harness evolution. Claude Opus 5.5 proposes variants of an agent harness for heuristic-optimization contests, each a complete harness that writes, tests, and refines a solution program. In the archive arm a variant descends from an

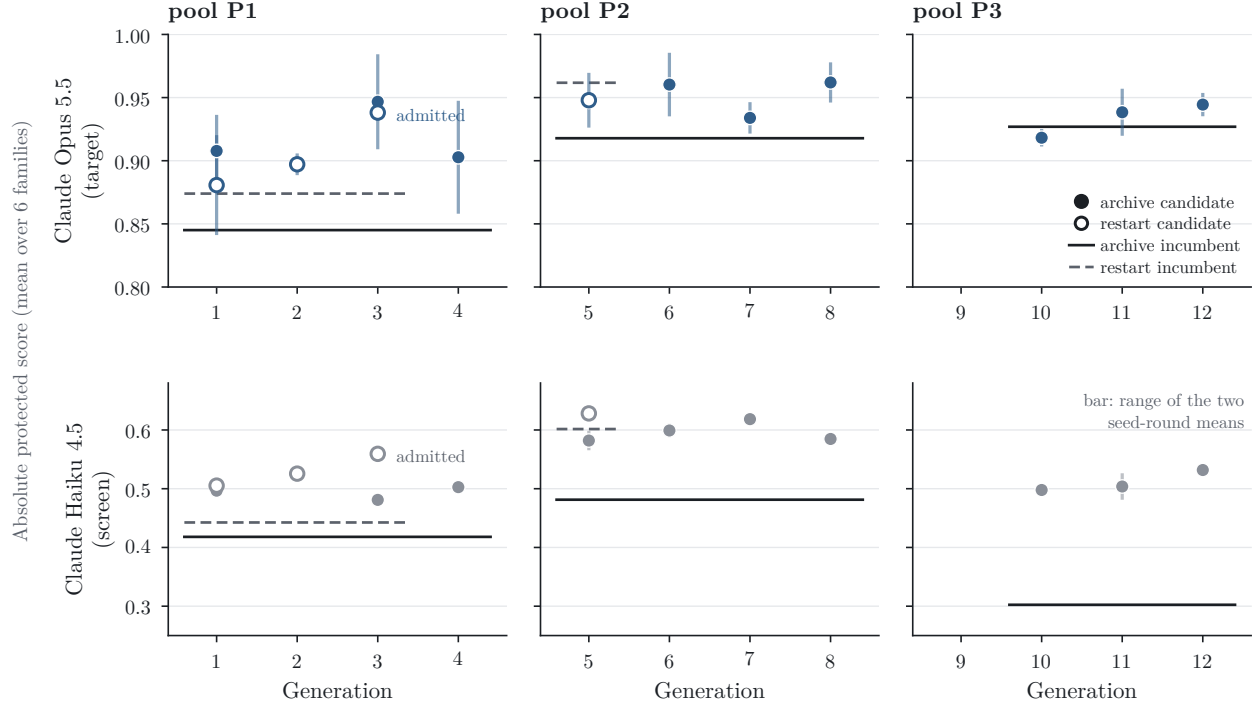


Figure 3: **Variants selected on a Claude Haiku 4.5 screen convert Claude Haiku 4.5’s residual error and leave Claude Opus 5.5’s where it was.** Absolute protected score of every screened harness variant (points) and of the incumbent harness of its time (lines), with Claude Opus 5.5 (top) and Claude Haiku 4.5 (bottom) as the solver, on each of the three protected problem pools. Each point is the mean over the pool’s six problem families of two seed rounds; bars span the two round means. Filled: archive arm; hollow: restart arm; the one variant the protected gate admitted is labeled. On the later pools the same variants sit far above Claude Haiku 4.5’s incumbent and within seed-round noise of Claude Opus 5.5’s.

archive of admitted variants; in the restart arm each generation starts afresh. Claude Haiku 4.5 is the screen: it scores each candidate on six selection problems with 16 proposals, and the loop selects on that score. A protected gate then measures each selected candidate’s gain over the incumbent of its time on normalized ALE-Bench score, with 32 proposals and an exact sign-flip test at $\alpha = 0.05$ split over two rounds, with Claude Haiku 4.5 and with Claude Opus 5.5 as the solver. The archive arm ran twelve generations of four proposals, one protected candidate per generation, and the loop was preregistered.

Self-improvement methods on one frontier model. Six of the ten arms of a grid on 16 ALE-Bench problems are reported, all with Claude Opus 5.5 at effort high, for every proposal and auxiliary call: ACE [26], ALE-Bench self-refine with eight parallel chains [14], a port of the Darwin Gödel Machine’s archive and parent selection [25], a port of the Huxley-Gödel Machine’s selection policy [20], a gated improver, and the best of 32 independent samples of the ALE-Bench prompt (bare). The other four arms are our own harnesses and variants, and none scored above ACE (2,442 to 2,647). Each arm ran once, seed 0, from a cold start with 32 proposal calls, eight in flight, and 64k maximum output tokens per proposal, and each final program received one official ALE-Bench private evaluation. The predictions were registered before any official evaluation.

5.2 Variants Selected on the Screen Convert the Screen’s Residual Error

The two solvers leave different residual errors on the same problems, and the screened variants convert only the screen’s (Figure 3). On the last pool the seed harness scored 0.30 with Claude Haiku 4.5 and 0.93 with Claude Opus 5.5. The three variants screened there beat it by 0.20 to 0.23 on Claude Haiku 4.5 and by -0.01 to $+0.02$ on Claude Opus 5.5. Across all three pools, the 14 screened variants scored 0.88 to 0.96 with Claude Opus 5.5, and only one cleared the protected gate. The Claude Haiku 4.5 selection score carries no information about the Claude Opus 5.5 gain: Spearman

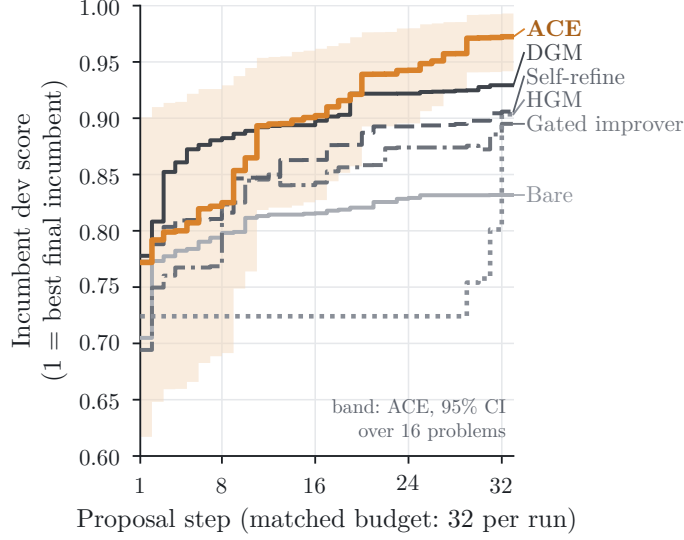


Figure 4: **ACE leads the other self-improvement methods on Claude Opus 5.5 at every step from step 20 of 32.** Mean normalized development score of each method’s incumbent (1 = best final incumbent on the problem) against proposal step, 16 ALE-Bench problems, matched at 32 proposal steps per run; band: ACE’s 95% interval over problems. ACE adds reflection and curation calls: 102 calls per run, against 32 to 40 for the other methods.

Table 9: **On Claude Opus 5.5, ACE ranks first among self-improvement methods, and the Huxley-Gödel Machine port ranks below bare sampling.** Mean official ALE-Bench performance over 16 problems, Claude Opus 5.5 at effort high for every call, matched at 32 proposal steps per run, one seed (seed 0), one official private evaluation per final program. Lift over bare with a paired bootstrap 95% interval over problems, and problems won/tied/lost against bare; calls and output tokens per run. Bold: best; underline: second.

Method on Claude Opus 5.5	Perf. $\uparrow$	Lift over bare [95% CI]	W/T/L	Calls	Out Mtok
ACE	2,738	+469 [+305, +639]	15/0/1	102	0.76
Self-refine (8 chains)	<u>2,548</u>	+279 [+140, +437]	12/0/4	33	0.42
Darwin Gödel Machine (port)	2,526	+257 [+119, +422]	13/0/3	33	0.55
Gated improver	2,508	+238 [+114, +373]	13/0/3	40	0.28
Bare (best of 32 samples)	2,269	–	–	34	0.44
Huxley-Gödel Machine (port)	2,217	–52 [–374, +202]	10/0/6	32	0.53

0.160 ($p = 0.58$, $n = 14$), and the Claude Haiku 4.5 gain against the Claude Opus 5.5 gain, -0.138 ($p = 0.64$). The screen selected what converts Claude Haiku 4.5’s failures, and on Claude Opus 5.5 those variants found almost nothing of that kind to convert.

A screen whose failures match the target’s keeps the ranking. Scored by Claude Opus 5.5, the mean screen score of nine self-improvement systems tracks their mean on unseen problems at Spearman 0.700; scored by Claude Haiku 4.5, across eight systems, at 0.476.

5.3 The Ranking of Self-Improvement Methods Follows the Base Model’s Failures

On Claude Opus 5.5 the six methods rank ACE (2,738) above self-refine (2,548), the Darwin Gödel Machine port (2,526), the gated improver (2,508), bare sampling (2,269), and the Huxley-Gödel Machine port (2,217) (Table 9). ACE beats bare sampling on 15 of 16 problems and the gated improver on 14, by 230 points [115, 354], with 2.6 times the gated improver’s calls and 2.7 times its output tokens; it leads on the development score at every step from step 20 of 32 (Figure 4). The Huxley-Gödel Machine reports ranking above the Darwin Gödel Machine on a 60-task SWE-bench Verified subset, 56.7 to 53.3 [20]; on Claude Opus 5.5 its port finishes 309 points below the Darwin Gödel Machine port and 52 below bare sampling.

The order changes with the model. On fresh ALE families scored by sealed area under the curve, the gated improver

beats ACE on gpt-oss-20b by $+0.088$ [$+0.020$, $+0.165$] over 50 problem-seed pairs, while on Claude Opus 5.5, on the same metric, ACE leads the gated improver by $+0.018$ [-0.015 , $+0.062$] over 30. On Claude Haiku 4.5 a restart arm beats it by $+0.112$ [$+0.032$, $+0.214$] over 30. Multi-agent search shows the same pattern. CORAL reports that co-evolving agents beat the best of four independent agents on Anthropic’s performance take-home, 1,103 cycles against 1,180 with Claude Opus 4.6, run to convergence [17]. In a matched 8-hour race on Claude Opus 5.5 at medium effort, with four agents per arm, three seeds, the same machine class, and the same evaluator, our CORAL-faithful port and the best of four independent agents finish at median 948 and 946 cycles. A gain measured on one model does not keep its size or its rank on another, as Corollary 2.2 requires when the models’ failures differ in cause.

What Claude Opus 5.5 fails on. The ranking follows from the causes of Claude Opus 5.5’s residual error on these contests. Sixteen candidate technique cards were each tested on Claude Opus 5.5 under an admission test on the six selection problems. Three were admitted, each at $p = 0.0001$, and all three state a problem’s key idea, an exact sub-method from the contest’s editorial. Every card of generic search craft failed, among them late acceptance, iterated local search, ALNS, NRPA, and layered beams, and two harmed. ACE’s playbook carries key ideas from one batch to the next: learning from scratch on AtCoder Heuristic Contest 034, it took Claude Opus 5.5’s programs from 0 to 24 of 32 using min-cost flow within four batches. Search craft converts the failures of a model that lacks it. Claude Opus 5.5 does not lack it, so a loop that supplies search craft has none of Claude Opus 5.5’s failures to convert.

5.4 An Improvement Record Supplies No Cause the Target Lacks

The Darwin Gödel Machine assumes that an agent that scores better is also a better improver of itself [25], and archive-based loops carry a growing archive of earlier discoveries forward [7]. Under the law a record helps a new problem only if it supplies a cause of that problem’s failures. On ten fresh ALE families with three seeds, Claude Opus 5.5’s gated improver inheriting the record it built on 21 training families gained $+0.020$ [-0.010 , $+0.060$] in sealed area under the curve over the same improver started cold, and $+0.003$ [-0.025 , $+0.032$] over a placebo record of the same size with its attribution scrambled. In a post hoc contrast, the scrambled placebo beat the cold improver by $+0.017$ [$+0.002$, $+0.038$]. The inherited record beat the placebo on none of three models: by $+0.016$ [-0.026 , $+0.061$] on gpt-oss-20b over 50 pairs and by $+0.045$ [-0.017 , $+0.129$] on Claude Haiku 4.5 over 30. What a record adds is the variants it holds; which improvement produced which gain adds nothing measurable.

6 Related Work

Recursive self-improvement of harnesses. ADAS has a meta agent program new agents in code from an archive of earlier discoveries [7], and the Darwin and Huxley-Gödel Machines grow archives of self-modifying coding agents [20, 25]. The Huxley-Gödel Machine chooses which agent to expand by Thompson sampling on its clade’s aggregate benchmark success, and it ranks itself above the Darwin Gödel Machine on a 60-task SWE-bench Verified subset, 56.7 to 53.3. On Claude Opus 5.5 over 16 ALE-Bench problems the order reverses, 2,526 for the Darwin Gödel Machine port and 2,217 for the Huxley-Gödel Machine port, below bare sampling (Table 9). The 2026 harness loops evolve prompts, control flow, tools, and memory around a frozen model. RRSI, Harness², and HarnessPro report end-score gains on the model they ran [16, 22, 23]. Prime Agent keeps a continual harness of prompt notes, memories, skills, and subagent specifications refined from its own trajectories [11]. MILO co-evolves harnesses for software-engineering agents and observes that harness gains “need not transfer across backbones” [9]. HarnessTTT trains a harness proposer at test time against a frozen executor and transfers the result to a stronger one [13], and Hill Sampling keeps only the best verified program [2]. Our program paper reports the headroom law, which it calls the headroom relation, with its fit on Claude Opus 5.5 and its out-of-sample forecast of Claude Opus 5.5 from weaker models [4]. This paper reports both in full. It also derives the consequence for loops that select on one model and deploy on another, which is the configuration of the Huxley-Gödel Machine’s GPT-5-mini evaluator and of HarnessTTT’s transfer.

Measuring what a harness adds. The Last AI Built by Humans normalizes released models’ scores into a Headroom-Closed Index and illustrates, without a fit, self-improvement closing 0.78 of each domain’s remaining headroom [5]. BiPA replays each optimizer’s acceptance rule on candidates identical to the seed and, on test, finds learned skills 0.1 points above the hand-written seed [28]. Its own Table 1 has the proportional form: against the agent with no skill, the seed recovers 0.10 to 0.14 of the residual error in five of six model-benchmark cells, by our arithmetic. Finding the Right Fit runs 66 configurations, 22 harness-model pairings on three task collections, and finds that the best harness

changes with the model [12]. When Agents Fail collects 52,168 trajectories and scores each pairing, on a balanced panel, by a Model-Harness Fit relative to the best observed harness [27]. Evolving to Match Capabilities finds that a skill’s benefit tracks the executor’s capability, with 86% of a weak executor’s failed runs violating a requirement already in the skill [24]. ACE’s own tables show the same reader dependence: on AppWorld, in its offline setting with labels, it recovers 0.30 of the residual error with DeepSeek-V3.1 (42.4 to 59.4) and 0.09 with GPT-OSS-120B (34.6 to 40.5), by our arithmetic [26]. Proposition 2.1 gives the reason: a skill supplies the cause, and the reader sets p_j . A component that restores state rather than describing it, as the recovery harness does for Claude Haiku 4.5 in Section 4.3, does not depend on the reader to convert.

Scaling laws and forecasting. Scaling laws forecast a model’s loss from its parameters, data, and compute [6, 10], and benchmark performance from training compute, BIG-Bench Hard to 6 points of mean absolute error one order of magnitude out [15]. Observational scaling laws predict downstream performance from a low-dimensional capability vector, and they forecast the gains of chain-of-thought and self-consistency at scale from fits on weak models [18]. The headroom law forecasts the lift of an agent-harness component over the same model without it, from that model’s residual error on the instrument, a quantity measured on the control before the component runs.

Weak-to-strong generalization. Weak-to-strong supervision trains a strong model on a weak model’s labels [3]. The forecast here is weak-to-strong in measurement: a share estimated on weaker models predicts the stronger model’s gain, with no training.

7 What the Law Forecasts Next

The law forecasts the next model generation on the same instruments before it runs. On instruments the model completes unaided, the residual error falls and every component’s lift falls in proportion. On a handover to Qwen3.5-72B, an API whose meaning changes without notice, and scheduled crashes, the cause lies outside the episode, and the residual error and the lift persist at a share near 0.54. The next experiment holds each component and instrument fixed, replaces Claude Opus 5.5 with its successor, classifies the control’s failures by cause from their traces before any arm runs, and compares the measured share with $\sum_j \phi_j p_j$ predicted from those traces.

8 Conclusion

A harness component’s lift is a share of the residual error its model leaves, and the share is the fraction of that error whose cause the component supplies. Fitted on Claude Haiku 4.5 and Claude Opus 4.8, the share forecasts Claude Opus 5.5’s gains better than any gain that ignores the residual error, and the sensor that rescued all 32 tasks an API change broke, and none of the 9 failures the change did not cause, shows why. The same arithmetic explains why self-improvement selected on a weaker screen converts that screen’s failures and loses its rank on the target. A loop that improves a harness for a frontier model must screen on that model, or on a model whose failures have the same causes.

Acknowledgments

We thank the maintainers of AppWorld, Terminal-Bench, τ^2 -bench, and ALE-Bench, and Anthropic for its performance take-home.

References

- [1] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2 -Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv preprint arXiv:2506.07982*, 2025.
- [2] Jacob Beck, Philip V. Ogren, and Ari Kobren. Hill Sampling for Test-Time Scaling: A Simple and Better Alternative to Repeated Sampling, Evolution, and Training. *arXiv preprint arXiv:2609.25510*, 2026.
- [3] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision. *arXiv preprint arXiv:2312.09390*, 2023.

- [4] Lindell S. Cumes. Evidence-Gated Recursion: An Experiment on a Self-Governing Research Organization of Frontier Language-Model Agents. QWNTL Labs; alphaXiv 2610.nenya-headroom-law-recursive-harnesses (v2), <https://qwntl.com/frontier>, 2026.
- [5] Yi Duan, Ying Liu, Zirui Tang, Haodong Chen, Jun Zhou, Yumou Liu, Bangrui Xu, Yukai Wu, Sidi Chen, Yuhan Zhou, Haoyu Wang, Xiaoyou Yu, Shaokun Han, Xuzhou Zhu, Le Zhou, Bolin Lu, Wei Zhou, Jiachen Liu, Nuozhou Fang, Jiaxin Tian, Ruoyu Chen, Yuxuan Li, Kai Zuo, Kaiyan Zhang, Qianyu Yang, Zijie Wang, Jiantao Qiu, Conghui He, Guoliang Li, Bowen Zhou, Zhiyuan Liu, Zhoufutu Wen, Jihua Kang, Xuanhe Zhou, and Fan Wu. The Last AI Built by Humans: Toward Genuine Recursive Self-Improvement. *arXiv preprint arXiv:2609.11873*, 2026. URL <https://arxiv.org/abs/2609.11873>.
- [6] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556*, 2022. URL <https://arxiv.org/abs/2203.15556>.
- [7] Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems. *arXiv preprint arXiv:2408.08435*, 2024.
- [8] Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. ALE-Bench: A Benchmark for Long-Horizon Objective-Driven Algorithm Engineering. *arXiv preprint arXiv:2506.09050*, 2025. NeurIPS 2025 Datasets and Benchmarks Track.
- [9] Prithwish Jana, Mononito Goswami, Hao Liu, Xinyu Li, Langlin Huang, Zhehui Huang, Zhishen Huang, Patrick Blöbaum, Anoop Deoras, Purak Jain, Nikos Kanakaris, and Sahika Genc. MILO: Automated Harness Discovery via Orchestrated Multi-Agent Evolution. *arXiv preprint arXiv:2609.38349*, 2026. Project page: <https://jprithwish.github.io/MILO/>.
- [10] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*, 2020. URL <https://arxiv.org/abs/2001.08361>.
- [11] Seth Karten, Alex L. Zhang, Kevin Thomas, Sebastian Müller, Elie Bakouch, Daniel Auras, Mika Senghaas, Fares Obeid, Konstantin Dunas, Johannes Hagemann, and Sami Jaghouar. Prime Agent: A Self-Improving RLM Harness. *arXiv preprint arXiv:2608.23552*, 2026. URL <https://arxiv.org/abs/2608.23552>.
- [12] Yixuan Li, Yiyun Zhou, Yao Long Teng, Fuchao Yang, Yanchen Deng, Zhiyi Lyu, Xuyu Dong, Feng Chen, and Bo An. Finding the Right Fit: Model-Harness Interactions across Agent Tasks. *arXiv preprint arXiv:2610.00917*, 2026. URL <https://arxiv.org/abs/2610.00917>.
- [13] Yingzi Ma, Yulong Cao, Nanxi Li, Fang Wu, Yizhe Zhang, James Zou, Yejin Choi, and Chaowei Xiao. HarnessTTT: Learn to Optimize Harness at Test-Time for Scientific Discovery. *arXiv preprint arXiv:2610.17370*, 2026.
- [14] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv preprint arXiv:2303.17651*, 2023.
- [15] David Owen. How predictable is language model benchmark performance? *arXiv preprint arXiv:2401.04757*, 2024. URL <https://arxiv.org/abs/2401.04757>.
- [16] Zhuoshi Pan, ZhiYuan Feng, Yukuo Cen, Honglin Lin, Junru Lu, Zhongxia Chen, Shuo Yang, H. Vicky Zhao, and Jie Tang. HarnessPro: Recursive Harness Self-Improvement with Pareto Optimization for Performance-Cost Trade-offs. *arXiv preprint arXiv:2610.26770*, 2026. URL <https://arxiv.org/abs/2610.26770>.
- [17] Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, Jiacheng Zhu, Xuan Jiang, Sirui Li, Cathy Wu, Bryan Kian Hsiang Low, Jinhua Zhao, and Paul Pu Liang. CORAL: Towards Autonomous Multi-Agent Evolution for Open-Ended Discovery. *arXiv*

- preprint arXiv:2604.01658*, 2026. COLM 2026 (per the authors’ repository). Code: <https://github.com/Human-Agent-Society/CORAL>.
- [18] Yangjun Ruan, Chris J. Maddison, and Tatsunori Hashimoto. Observational Scaling Laws and the Predictability of Language Model Performance. *arXiv preprint arXiv:2405.10938*, 2024. URL <https://arxiv.org/abs/2405.10938>. arXiv comment: Accepted at NeurIPS 2024 as a spotlight.
 - [19] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. *arXiv preprint arXiv:2407.18901*, 2024. ACL 2024.
 - [20] Wenyi Wang, Piotr Piękos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel Machine: Human-Level Coding Agent Development by an Approximation of the Optimal Self-Improving Machine. *arXiv preprint arXiv:2510.21614*, 2025.
 - [21] Zimu Wang, Xiaobo Wang, Siyu Ye, Jiaqi Li, Silin Chen, Yuling Shi, Ruobing Wang, Jingjing Zhang, Chaofan Wang, Kai Wu, Mengnan Qi, Kai Cai, Xiaodong Gu, and Zilong Zheng. RSIHub: A Composable Framework for Agent-Evolution Methods. *arXiv preprint arXiv:2610.32253*, 2026. URL <https://arxiv.org/abs/2610.32253>. arXiv comment: NeurIPS 2026 Meta-Agents Workshop.
 - [22] Peng Xia, Rujun Han, Zifeng Wang, Yanfei Chen, Yufan Zhuang, Yoonho Lee, Chengsong Huang, Han Yu, Zhongying CuiZhu, Yifei Ming, Huaxiu Yao, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. RRSI: Regularized Recursive Self-Improvement of Agent Harnesses. *arXiv preprint arXiv:2609.24972*, 2026. URL <https://arxiv.org/abs/2609.24972>.
 - [23] Ruiyao Xu, Yanfei Chen, Zhongying CuiZhu, Bhavana Dalvi Mishra, Yifei Ming, Han Yu, Rujun Han, Chen-Yu Lee, and Tomas Pfister. Harness²: Recursive Agent Harnessing for an Open World. *arXiv preprint arXiv:2610.14224*, 2026. URL <https://arxiv.org/abs/2610.14224>.
 - [24] Zhihao Yang, Haojin Wang, Juncheng Wu, Kuan Zhang, Yukun Chen, Run Luo, Yaojie Lu, Yuyin Zhou, Lu Wang, and Xianpei Han. Evolving to Match Capabilities: Improving Harness Self-Evolution via Skill Adaptation. *arXiv preprint arXiv:2610.33024*, 2026. URL <https://arxiv.org/abs/2610.33024>.
 - [25] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv preprint arXiv:2505.22954*, 2025.
 - [26] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv preprint arXiv:2510.04618*, 2025. ICLR 2026.
 - [27] Shao Zhang, Hangfan Zhang, Yuanyuan Shi, Hao Li, Min-Le Su, Kangcong Li, Jingping Wang, Chunjiang Mu, Chen Zhang, Yang Chen, Bo Zhang, and Shuyue Hu. When Agents Fail: Model or Harness—Who’s to Blame? *arXiv preprint arXiv:2610.32483*, 2026. URL <https://arxiv.org/abs/2610.32483>.
 - [28] Jiawei Zhou, Bo Peng, Yuxuan Huang, Ming Zhang, Youbang Sun, Xiang Wang, and Chaochao Lu. Does Your Generalist Skill Optimizer Really Incentivize Agentic Tasks? Rethinking Skill Optimization via Bipartite Attribution. *arXiv preprint arXiv:2610.37201*, 2026. URL <https://arxiv.org/abs/2610.37201>.

A Alternative Models and the Cross-Model Test

Table 10 fits the law and five alternatives to the 15 Claude Opus 5.5 comparisons and gives, for each alternative, the Bayes factor of the weak-to-strong forecast (Definition 2.3). The smooth cap rises with slope 1 at small residual error and saturates at c . The law with a shift gains AIC on the Claude Opus 5.5 comparisons and loses leave-one-out score, the mark of a parameter that fits noise. Fitted on the six weaker comparisons its shift is 10.8 points, and the law without it forecasts Claude Opus 5.5 with a Bayes factor of 13.7 against it.

Table 10: **The law against five alternatives.** Columns 2–6: each model fitted on the 15 Claude Opus 5.5 comparisons; LOO is the leave-one-out predictive log score (higher is better). Last column: Bayes factor of the law against the model when both are fitted on the six Claude Haiku 4.5 and Claude Opus 4.8 comparisons and scored on the 15 Claude Opus 5.5 comparisons, each with its own fitted spread.

Model	Mean lift μ_i (fit on Claude Opus 5.5)	τ	AIC	BIC	LOO	Forecast BF
No effect	0	7.09	107.10	107.81	−53.21	186
Constant gain	$c = 5.71$	4.16	96.82	98.24	−48.30	9.8
Capped constant gain	$\min(c, h), c = 7.33$	2.64	89.39	90.81	−45.37	5.6
Smooth cap	$c(1 - e^{-h/c}), c = 12.1$	1.89	83.15	84.57	−42.08	3.1
Law with a shift	$kh - b, k = 0.641, b = 1.59$	0.00	75.58	77.70	−41.14	13.7
Headroom law	$kh, k = 0.539$	0.98	77.16	78.57	−38.58	–

B Sensitivity of the Fit to Row Selection

Table 11 refits Equation (2) after removing one group of rows at a time, and then all of them. The four carried-knowledge rows are L010–L012 and L016; L124 is a subset of L126; L057 is the one mid-task handoff; L099 was not independently recomputed and L166 was analyzed after the fact; L174 mixes Claude Opus 4.8 and Claude Opus 5.5. The share stays between 0.50 and 0.57 under each single removal and is 0.50 with all six groups removed, and the law has the lower AIC on each row set, by 6.3 to 19.7 points.

Table 11: **Fits after removing groups of comparisons** (Claude Opus 5.5 comparisons against a no-harness control; profile-likelihood 95% intervals with τ re-optimized at each point).

Row set	n	k [95% CI]	AIC, constant / law
All comparisons	15	0.539 [0.427, 0.675]	96.8 / 77.2
Without the four carried-knowledge rows	11	0.563 [0.443, 0.713]	68.1 / 55.5
Without a row that is a subset of another	14	0.514 [0.400, 0.655]	90.1 / 72.0
Without the one mid-task handoff	14	0.537 [0.422, 0.677]	89.3 / 71.5
Without one row not independently recomputed and one post hoc row	13	0.499 [0.383, 0.643]	82.9 / 65.9
Without the mixed-model AppWorld stream	14	0.568 [0.440, 0.718]	91.9 / 73.3
Without all of the above	6	0.503 [0.338, 0.748]	36.5 / 30.1

C Fixed Texts

The diligence instruction on AppWorld is the following four sentences, appended to the standard prompt: “Working principles (general, not specific to any app): be thorough and verify against the live app state. When an API returns an empty, missing or surprising result where you expected data, do not take it at face value: investigate before concluding, check the arguments you passed, and try reasonable alternatives. Make sure you have collected all of the data the task needs. Before calling `complete_task`, re-check that your answer and every required action are correct and complete.” The crash notice given to the bare model told of a crash states that the agent process crashed and was restarted, that the conversation could not be restored, how many steps had run before the crash, that the app state persists and every API call that ran before the crash took effect, and that the Python session was restarted.

D The Weaker-Model Comparisons and Per-Comparison Forecasts

Table 12 lists the six comparisons the forecast is fitted on, and Table 13 the forecast for each Claude Opus 5.5 comparison.

Table 12: **The six weaker-model comparisons the forecast is fitted on.** Points on a 0–100 scale, as recorded and fitted. L058’s recorded standard error, 4.92, corresponds to all 168 tasks of its stream; at the 84 tasks of the contrast it is 6.96, and refitting with it raises the forecast’s Bayes factors to 958, 26.3, and 12.2.

Row	Model	Component against control; instrument (n)	Control	h	Δ (s)	r
L054	Claude Opus 4.8	Handoff brief from Claude Opus 5.5 mid-task against Claude Opus 4.8 alone; AppWorld challenge (66)	90.9	9.1	6.10 (4.12)	0.67
L002	Claude Haiku 4.5	Store compiled by Claude Opus 5.5 against bare; AppWorld test-normal TGC (168)	80.1	19.9	6.50 (4.05)	0.33
L058	Claude Haiku 4.5	Its own online store against none; second half of a 168-task stream (84)	78.6	21.4	−17.90 (4.92)	−0.84
L007	Claude Haiku 4.5	Recovery harness against bare told of the crash; AppWorld fresh normal (58)	72.4	27.6	17.24 (5.28)	0.62
L001	Claude Haiku 4.5	Store compiled by Claude Opus 5.5 against bare; AppWorld test-challenge TGC (417)	66.6	33.4	8.90 (3.13)	0.27
L006	Claude Haiku 4.5	Recovery harness against bare told of the crash; AppWorld challenge (65)	63.1	36.9	29.23 (5.89)	0.79

Table 13: **Forecast of each Claude Opus 5.5 comparison** from the fits on the six weaker comparisons: the law ($0.383h$), the capped constant gain ($\min(8.12, h)$), and the constant gain (8.12), against the measured lift. Points on a 0–100 scale.

Row	h	Measured Δ	Law	Capped	Constant
L010	0.0	−1.10	0.00	0.00	8.12
L003	2.3	0.00	0.88	2.30	8.12
L016	5.0	−1.67	1.92	5.00	8.12
L004	8.0	5.75	3.07	8.00	8.12
L136	9.7	2.68	3.72	8.12	8.12
L011	10.0	0.00	3.83	8.12	8.12
L163	10.1	4.30	3.87	8.12	8.12
L012	11.0	1.40	4.22	8.12	8.12
L099	11.0	10.80	4.22	8.12	8.12
L147	13.1	9.30	5.02	8.12	8.12
L174	15.1	6.70	5.79	8.12	8.12
L126	15.6	10.80	5.98	8.12	8.12
L124	16.7	11.90	6.40	8.12	8.12
L166	19.3	13.20	7.40	8.12	8.12
L057	25.8	15.20	9.89	8.12	8.12
MAE			3.30	4.02	5.17

E Self-Improvement Methods on Claude Opus 5.5, per Problem

Table 14 gives the provenance of each rival method as run here. Each is a port to ALE-Bench’s heuristic contests from the authors’ public code; none of the authors’ own benchmarks was rerun in this harness.

Table 14: **Ports of rival self-improvement methods.** Source commit, implementation, and what the port changes to run on ALE-Bench with Claude Opus 5.5.

Method	Source	Prompts	Changes for ALE-Bench
ACE	ace-agent/ace at 82709de; arXiv 2510.04618v3	Re-implemented Generator, Reflector, Curator	C++ generator graded on development execution; one reflection per proposal; no embedding deduplication
Darwin Gödel Machine	jennyzt/dgm at a565fd2	Archive and parent selection ported; self-modification dialogue not copied	One complete-program edit per proposal; selection by mean relative development score
Huxley-Gödel Machine	metauto-ai/HGM at 013872d	Reuses the DGM port’s proposal prompt	Clade-metaproductivity policy with $\alpha = 0.6$ and cooling, as in the paper; budget in proposal calls
Self-refine	SakanaAI/ALE-Bench at 3da9b12	Reference wording adapted to C++20	Eight independent parallel chains

Table 15 gives the official ALE-Bench performance of each method’s final program on each of the 16 problems. Eight problems were preregistered and eight added in a preregistered extension; every arm ran cold, seed 0, with Claude Opus 5.5 at effort high for every call.

Table 15: **Official ALE-Bench performance per problem on Claude Opus 5.5** (higher is better), one private evaluation per final program, 32 proposals per run. DGM and HGM are ports of the Darwin and Huxley-Gödel Machines’ selection rules. Bold: best in each row.

Problem	ACE	Self-refine	DGM	Gated improver	Bare	HGM
ahc001	2,114	2,473	2,392	2,301	2,023	1,978
ahc003	3,195	2,477	2,925	2,630	2,537	564
ahc007	2,960	3,027	3,497	2,960	2,338	2,960
ahc008	1,656	1,677	1,630	1,638	1,591	1,744
ahc010	2,696	2,529	2,476	2,319	2,067	2,510
ahc011	2,806	2,631	2,655	2,588	2,455	2,550
ahc014	2,317	2,317	2,307	2,307	2,329	2,237
ahc015	2,882	2,400	2,652	2,652	2,005	2,302
ahc016	3,517	2,898	2,650	2,650	2,480	2,995
ahc025	3,228	2,259	2,722	2,966	2,308	2,398
ahc026	2,940	3,001	2,678	2,849	2,637	1,733
ahc034	3,113	2,385	2,148	2,814	2,110	2,212
ahc038	2,260	2,306	2,194	2,213	2,090	2,194
ahc041	2,964	2,556	2,544	2,610	2,625	2,443
ahc045	2,434	2,356	2,226	2,200	2,334	2,226
future-contest-2022-qual	2,728	3,483	2,728	2,427	2,379	2,427
Mean	2,738	2,548	2,526	2,508	2,269	2,217