Move42: InnovationZero Learns to Invent Executable Algorithms

A Verifier-Driven Foundation Model from Data to Code

Jepson Taylor, Jordan Hildebrandt, Chris Brousseau, Alton Alexander, and Kelli Quinn

***Abstract*—Most foundation models map a dataset and an input to a prediction. We study a model whose primary output is instead a standalone learning algorithm. Move42 is the research program; InnovationZero is its data-to-code foundation model. It represents a training problem, proposes one grammar-valid program genome, compiles that genome, and emits executable code whose utility is measured by an independent referee. The operational thesis is that repeated, independently scored program construction can be compiled into a policy that emits a useful first algorithm for a new dataset. Under the strict first-move protocol, with no post-evaluation choice among candidates, InnovationZero obtains 1108.8 Elo and rank 14/98 in the sealed field. Bounded search and retained-frontier diagnostics obtain 1212.2 and 1331.1 Elo, respectively, but answer separate questions. We formalize the game, legal and forbidden moves, compiler contract, evaluator, training receipts, and the supervised/ranking losses that convert external measurements into gradients without differentiating through code or held-out RMSE. We also receipt two general-agent plays and release a post-hoc discovery atlas of exactly 600 outcome-ranked cases: 353 wins and 247 losses against TabFM Ensemble. That split is atlas composition, not a prospective success probability. Margin distributions, matched failures, and three mechanism studies show the narrower result: generated representations can expose useful inductive biases, but the same motifs also fail when mismatched to data.**

***Index Terms*—algorithm discovery, foundation models, program synthesis, verifier feedback, tabular learning, data-to-code**

I. Prediction Versus Algorithm Invention

A fixed predictor implements

$$f_\phi(D, x) \longmapsto \hat{y}, \tag{1}$$

where pretraining selects a family of answers. InnovationZero instead implements

$$D \longmapsto a_D, \qquad a_D : (X, y, x) \longmapsto \hat{y}, \tag{2}$$

where a_D is executable code. This output is richer than a prediction because it may introduce a representation, kernel, interaction, regularizer, or estimation procedure. The distinction is also testable: the code can be frozen, inspected, run without the proposal model, and independently scored.

We use *innovation* in a deliberately falsifiable sense. An artifact must be dataset-conditioned, legal under a declared grammar, structurally distinct under canonical program identity, executable under the local contract, and independently useful. This does not assert consciousness, autonomous science, causal explanation, or fundamentally new mathematics. It asserts the creation of a new program artifact relative to the retained lineage and its evidence record.

The central scientific question is not whether unlimited search eventually finds a good pipeline. It is whether past games can be amortized into a useful first proposal. For proposal policy π_θ and evaluator U,

$$a_N^*(D) = \arg \max_{a \sim \pi_\theta(\cdot|D),\, 1 \ldots N} U(a, D). \tag{3}$$

Strict first move fixes $N = 1$: one legal artifact is emitted and there is no outcome-based tournament among attempts. Bounded and frontier modes remain informative, but cannot be relabeled as first move.

We ask five questions:

1) Can accumulated game experience improve the first legal proposal?
2) What makes a proposal legal, fair, executable, and independently scoreable?
3) How does experimental feedback train a neural model without differentiating through the evaluator?
4) How do general-purpose OpenAI and Anthropic agents perform when forced to construct one fixed legal contender?
5) In the released atlas, which structures accompany wins and losses, and what does that reveal about matching algorithms to data?

The evidence is intentionally separated. The strict result tests amortized first-proposal quality. The 600 cases were ordered using their observed RMSE ratio and are released for discovery and failure inspection. Their 58.83% win composition is not a blind, representative, or population win-rate estimate.

II. A Foundation Model from Data to Code

Figure 1 gives the high-level architecture:

$$\begin{aligned} \text{table} &\to \text{dataset/target representation} \\ &\to \text{proposal policy} \to \text{genome} \\ &\to \text{compiler} \to \text{code}. \end{aligned} \tag{4}$$

The representation conditions invention-time decoding. The final artifact is self-contained: it cannot call InnovationZero, an LLM, TabFM, TabPFN, TabICL, a hosted service, a downloadable checkpoint, or a model zoo. Foundation representations can therefore help choose a construction while remaining absent from deployment.

The policy emits a complete canonical genome rather than prose. Canonicalization removes representational aliases, permits exact deduplication, and binds a program to compiler

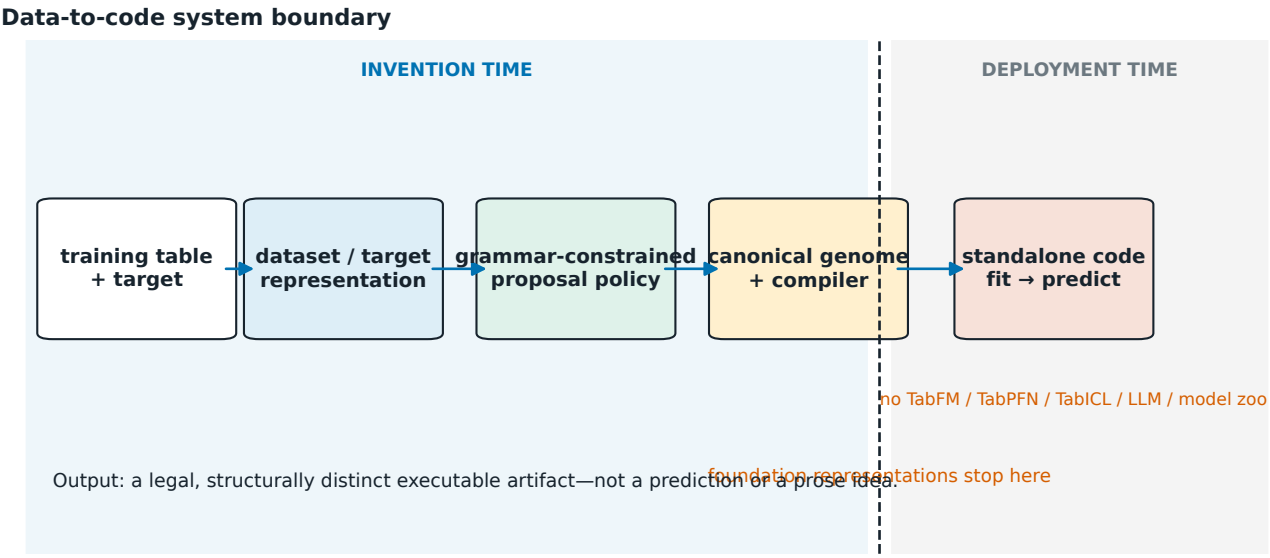

Fig. 1. Data-to-code boundary. The training table and target condition invention; the canonical genome compiles into standalone code. Deployment excludes foundation predictors, LLMs, services, and model-zoo lookup. Diagram fields are sourced by the game-specification receipt.

semantics. The compiler is deterministic; any primitive admitted by the grammar must have runtime parity. Dimensions, vocabulary allocation, reward heads, and compatibility loading are audit details rather than the conceptual definition and appear in Appendix A.

This design differs from choosing a named estimator. The output can compose bounded transforms and a terminal head, so two artifacts with the same head may implement materially different bases. Conversely, a familiar head does not make the whole construction trivial: the scientific object is the dataset-conditioned pipeline and its independent receipt.

III. The Algorithm Game

We define a game instance as

$$\mathcal{G} = (D, S, \mathcal{A}, C, E, P), \tag{5}$$

where D is the dataset-side learning problem, S is permitted state, $\mathcal{A}$ is the program grammar, C is canonicalization and compilation, E is the independent evaluator, and P is retention and promotion.

A. State, action, transition, outcome

The policy-visible state includes a dataset representation and target summary. Experience rows may additionally carry the prior legal genome, train-side score, provenance, rank percentile, runtime, instability, and former-self metadata. Dataset hashes, filenames, report identities, held-out targets, and test residuals are not predictive inputs.

An action is one complete pipeline:

```
(pipe X <zero or more transforms>
<one terminal head>)
```

It is not advice about what another program should try. A transition decodes under the legal mask, validates, canonicalizes, compiles, refits every learned quantity fold-locally, scores, writes a receipt, ranks within the dataset, retains or rejects, and appends experience. A terminal outcome is a valid finite RMSE, an explicit failure, or missing coverage. Failure and missingness are never silently converted into ordinary finite losses.

B. Legal moves

One terminal head is mandatory and last. Featmath permits at most 6 generated transforms and NewMath at most 4. Feature width is capped at 512, sequence length at 128, and symbolic expressions at depth 4 with at most 15 nodes. Fitted state and target-aware transforms are learned inside training folds. Seeds are explicit, outputs must be finite, and compiler/runtime parity is a legality condition.

These restrictions serve scientific identification. A bounded language exposes what was searched; fold-local fitting blocks target leakage; finite outputs make failure observable; deterministic execution lets the receipt identify the action.

C. Forbidden moves

The following invalidate a play:

- test labels, test residuals, or selection based on held-out outcomes;
- joint fitting on training and test rows;
- dataset hashes, fingerprints, filenames, or report lookup as features;
- TabFM, TabPFN, TabICL, LLMs, hosted inference, or checkpoint downloads inside the artifact;
- general AutoML, unrestricted model zoos, or direct PriorLabs/TabPFN implementations;
- network calls, subprocess escape, nondeterministic or unbounded fit-time search, and evaluator modification;
- per-dataset replacement of a frozen contender after outcomes are observed.

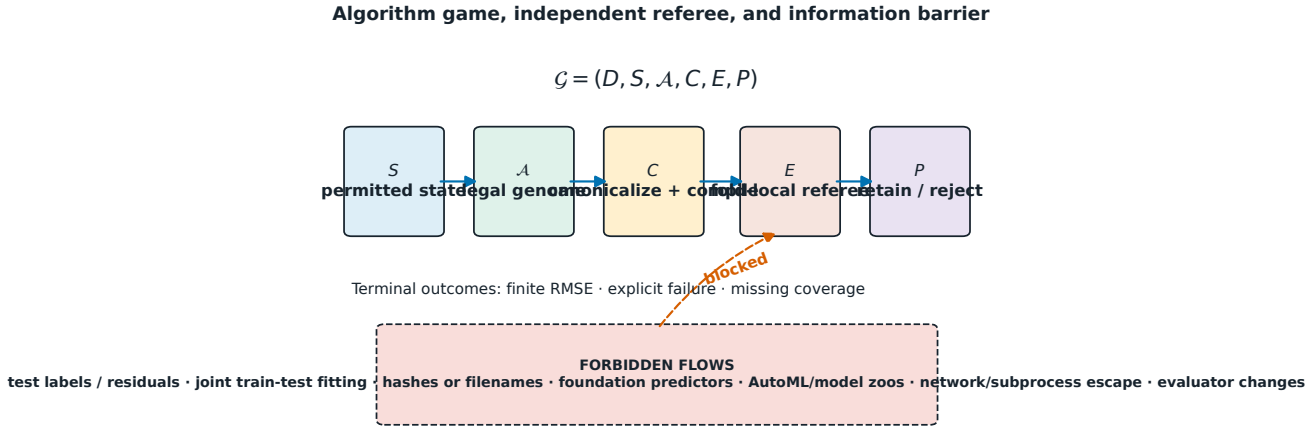

Fig. 2. Formal game and referee. Solid arrows are legal transitions; the dashed red path represents forbidden information. The evaluator sees a compiled artifact and fold-local data, not a proposal that can rewrite the referee after observing outcomes.

The reasons are not cosmetic. These moves respectively leak the answer, change the population, hide the actual dependency, erase the declared search budget, or move the goalposts.

IV. Referee and Executable Contract

The standalone interface is verbatim:

```
python3 model.py train.csv test.csv
predictions.csv
```

The training table contains numeric features followed by the target in its final column; the test table contains the same feature columns without the target. The output contains exactly one finite numeric prediction per test row, in stable input order. Sanitation and missing-value handling are deterministic. Preprocessing state is fit on training rows and applied unchanged to test rows. A malformed output, timeout, nonfinite value, wrong row count, or runtime error produces an explicit failure receipt.

For valid predictions,

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{6}$$

A lower RMSE wins; a higher RMSE loses; equality follows the scorer's numerical convention. Elo is meaningful only with its field, dataset protocol, artifact, and mode receipt. A bare Elo value is incomplete.

Strict, bounded, frontier, external-agent, and atlas modes are summarized in Table I and Fig. 3. The table is a protocol firewall: values are never pooled into a single maximum.

V. External Scores Become Neural Loss

The critical boundary is shown in Fig. 4. InnovationZero does not backpropagate through compiled programs, held-out RMSE, Elo, test predictions, or external foundation systems. Evaluation produces immutable scalars and ranks. Those labels supervise differentiable surrogate heads.

The complete cycle is: represent the problem; decode legal candidates; canonicalize and deduplicate; compile locally; refit and evaluate in train-side folds; compute a robust score; assign within-dataset ranks and former-self metadata; promote or reject; serialize experience; form masked minibatches; optimize the multi-task objective; and select a checkpoint using validation pairwise accuracy.

A. Measured training target

The robust train-side score is

$$s_{\mathrm{robust}} = \max(R^2_{\mathrm{OOF}}, -1) - 0.25\min\left(\mathrm{sd}(R^2_{\mathrm{folds}}), 1\right). \tag{7}$$

It clips catastrophic R^2 and penalizes fold dispersion. Nonfinite values remain missing and are masked from ranking.

B. Legal-set sequence imitation

At token t, illegal logits are masked before normalization. With legal set A_t, target y_t, and smoothing $\epsilon = 0.05$,

$$L_{\mathrm{seq},t} = (1-\epsilon)[-\log p_\theta(y_t)] + \epsilon\left[-\frac{1}{|A_t|}\sum_{a\in A_t}\log p_\theta(a)\right]. \tag{8}$$

Padding contributes nothing; forced single-choice positions have zero NLL. Per-row imitation can be reweighted by within-dataset advantage and provenance. An unscored legal row can still teach syntax without receiving a fabricated reward.

For scored rows in group G, reward-weighted imitation uses

$$A_i = s_i - \bar{s}_G, \quad \tau_G = \max(0.5\,\mathrm{sd}(s_G), 0.05), \tag{9}$$

and normalized weights proportional to $\exp(A_i/\tau_G)$ times the provenance multiplier.

C. Conditional ranking, uncertainty, and ablation

The reward branch interacts dataset context z with genome embedding g through

$$\phi(z, g) = [g,\ g \odot \gamma(z),\ |g - \mathrm{proj}(z)|]. \tag{10}$$

TABLE I
EVALUATION MODES AND THEIR NONINTERCHANGEABLE MEANINGS.

Mode	Artifact policy	Selection access	Scientific meaning
Strict first move	One dataset-conditioned artifact; 1108.8 Elo, rank 14/98	None after evaluation	Amortized first-proposal quality
Bounded probe	Fixed small budget; 1212.2 Elo, rank 7/98	Declared train-side rule	Value of limited selection
Frontier diagnostic	Retained/best candidate; 1331.1 Elo, rank 6/98	Diagnostic access	Candidate-set ceiling
OpenAI/Anthropic challenge	One frozen global artifact	No per-dataset replacement	General-agent construction
Released atlas	Outcome-ranked cases	Post-hoc descriptive	Mechanism and failure inspection

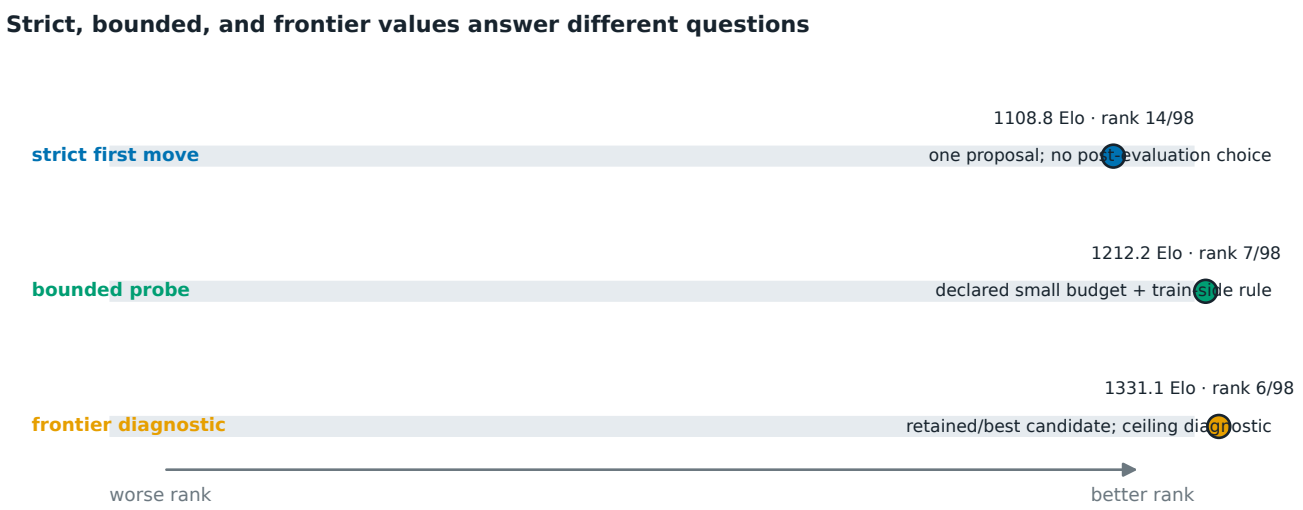

Fig. 3. Receipted field positions. Strict first move uses one artifact and no post-evaluation choice. Bounded selection and the retained frontier answer different scientific questions even when the field size matches.

4 deterministic bootstrap heads yield a mean reward and a disagreement-based uncertainty. A detached genome-only head receives the same pairwise ordering task but cannot use dataset context; conditional validation must outperform this ablation.

Within each dataset, finite unequal scores define a Bradley–Terry winner w and loser l:

$$L_{\mathrm{BT}} = \frac{1}{M} \sum_i w_i \,\mathrm{softplus}(r_{l_i} - r_{w_i} + m_i). \tag{11}$$

Weights depend on true score gap and are boosted for top-decile and former-self pairs. A former-self loser receives improvement margin 0.5. Deterministic reduction caps the batch at 4,096 pairs.

Groups with sufficient ranked rows also use a ListNet-style target at temperature 0.15:

$$\begin{aligned} q_i &= \mathrm{softmax}(p_i/0.15), \\ L_{\mathrm{list}} &= -\sum_i q_i \log \mathrm{softmax}(r)_i. \end{aligned} \tag{12}$$

Runtime uses Huber loss in $\log(1+c)$ space. Evaluation errors and negative raw OOF scores can train stability BCE. An optional family-classification CE preserves auxiliary representation information.

TABLE II
RECEIPTED COMPONENT SCALES.

Objective	Scale
Auxiliary CE	0.5
Genome-only ablation	1.0
Listwise reward	0.5
Pairwise reward	1.0
Runtime Huber	0.2
Sequence NLL	1.0
Stability BCE	0.5

Kendall–Gal uncertainty weighting is used for eligible multi-task components,

$$\exp(-s_j)\alpha_j L_j + \tfrac{1}{2} s_j, \tag{13}$$

while ranking objectives and auxiliary classification remain fixed-scale so they cannot suppress themselves. Learned log variances are clamped. Collapse controls include grammar masking, conditional-versus-genome-only validation, ranking in both phases, fixed ranking scales, former-self margins, uncertainty clamps, and checkpoint choice by validation pairwise accuracy.

Missing-label masks are semantic: absent scores do not become ranking loss, absent runtimes do not train the runtime

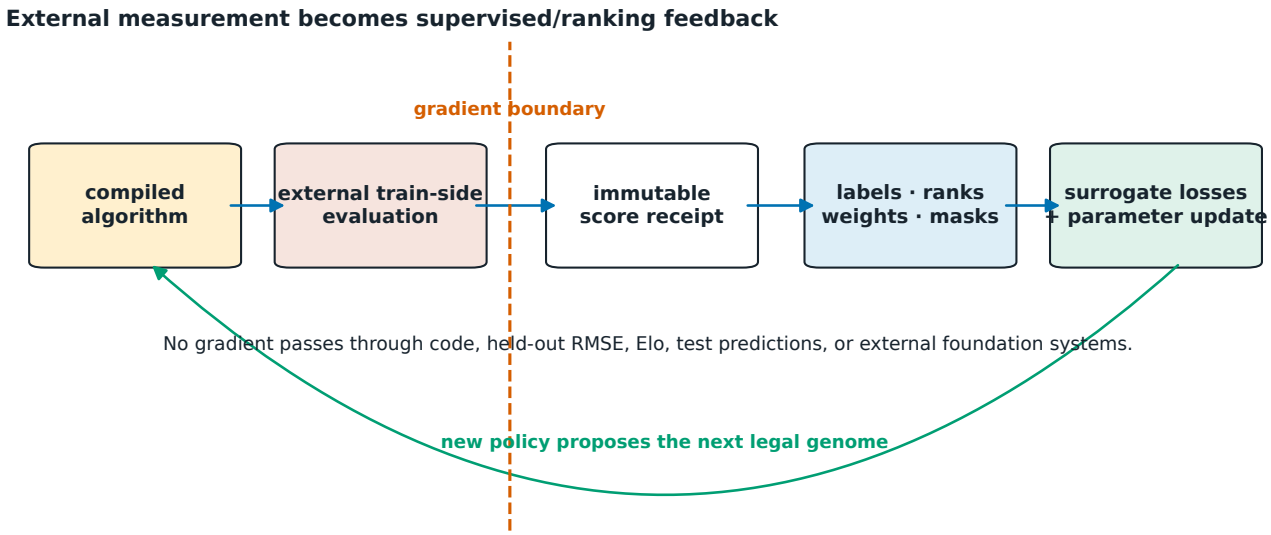

Fig. 4. Feedback boundary. Program execution and evaluation are outside the neural graph. Receipted labels, ranks, masks, and weights train the proposal model; no test outcome becomes a differentiable path.

head, and evaluation errors may train stability without becoming fabricated finite rewards.

VI. General Agents Play the Game

The external challenge asks a different question from InnovationZero first move: can a general agent construct one global deterministic contender? Both admitted examples obey one-frozen-artifact identity, hidden-dataset separation, no per-dataset replacement, and no foundation-predictor wrapper. Artifact hashes bind construction to evaluation.

A. OpenAI construction receipt

GPT-5.5 planning emitted `(pipe X (ensemble depth=3 rounds=150))`. One root agent, no subagents, and 124 tool calls produced the frozen artifact. Whole wall time was 56 min 24.533 s; planning was 6 min 47.324 s, implementation 49 min 20.361 s, and evaluation 2 min 7.249 s. Evaluation is a component of whole wall time, not an additive second lifecycle.

Provider accounting records 13,068,736 input tokens, including a 12,364,032 cached subset. Fresh input is 704,704 by subtraction. Output is 40,310, including 14,542 reasoning output; provider total is 13,109,046. Generated source length—3,578 Python tokens and 4,941 Rust tokens—is a separate resource axis. The fixed artifact scored 231 of the challenge cases with 1 unavailable, obtaining 1094.5 Elo and rank 18/98. The unavailable case remains missing coverage.

B. Anthropic construction receipt

The Anthropic run used one root `claude-fable-5` agent and one `claude-opus-4-8` exploration subagent, with 309 tool calls. Whole wall time was 1 h 45 min 34.141 s and evaluation 10 min 57.1 s. Provider-native categories are 176,007 fresh/input, 1,275,708 cache creation, 124,656,205 cache read, and 559,612 output. The receipt declares no single total, so we neither invent one nor compare cache-read tokens directly with OpenAI total tokens. The artifact scored 232 cases with 0 failures, obtaining 1298.3 Elo, 1297.2 double-holdout Elo, and rank 6/98.

GPT-5.6 workflow telemetry remains audit-only. No performance row is published because no receipt jointly binds model selector, session, artifact, protocol, coverage, score, and timestamps.

VII. The Released Discovery Atlas

The release contains exactly 600 unique case JSONs, ordered by the observed ratio against TabFM Ensemble. The signed margin is

$$m_i = \log\left(\frac{\mathrm{RMSE}_{\text{TabFM Ensemble},i}}{\mathrm{RMSE}_{InnovationZero,i}}\right). \tag{14}$$

Positive values are wins and negative values losses. The atlas contains 353 positive and 247 negative margins. The split and 58.83% percentage describe released composition only. We attach no confidence interval that would imply an unbiased population sample.

The median signed margin is 0.0059, and 190 cases lie within a one-percent multiplicative neighborhood of the boundary. Denominator sensitivity removes cases below fixed foundation-RMSE floors without reranking the remaining corpus. Search, final fit/predict, and total workflow time remain separate clocks.

A. Motifs, regimes, confidence failures, and duplicates

Figure 7 reports raw win/loss counts and within-atlas percentages by head and operator family, a train-row by feature-count regime map, train-side robust score against held-out margin, separated clocks, and CV dispersion. The point is contrast, not universal attribution: common motifs appear on both sides.

The source audit finds 503 source groups; 48 contain more than one released case and the largest contains 13. This

TABLE III
PROVIDER RECEIPTS PRESERVE ARTIFACT IDENTITY, CLOCKS, TOKEN SEMANTICS, AND SCORE AS DISTINCT AXES.

Field	OpenAI	Anthropic
Construction	GPT-5.5; one root; no subagents	claude-fable-5; one Opus exploration subagent
Artifact	DSL ensemble; fixed SHA-256 receipt	fixed global artifact; SHA-256 receipt
Wall / evaluation	56 min 24.533 s / 2 min 7.249 s	1 h 45 min 34.141 s / 10 min 57.1 s
Provider-native tokens	13,068,736 input (12,364,032 cached subset); 40,310 output	176,007 fresh; 1,275,708 cache-create; 124,656,205 cache-read; 559,612 output
Score	1094.5 Elo; rank 18/98; 231/232 scored	1298.3 Elo; rank 6/98; 232/232 scored

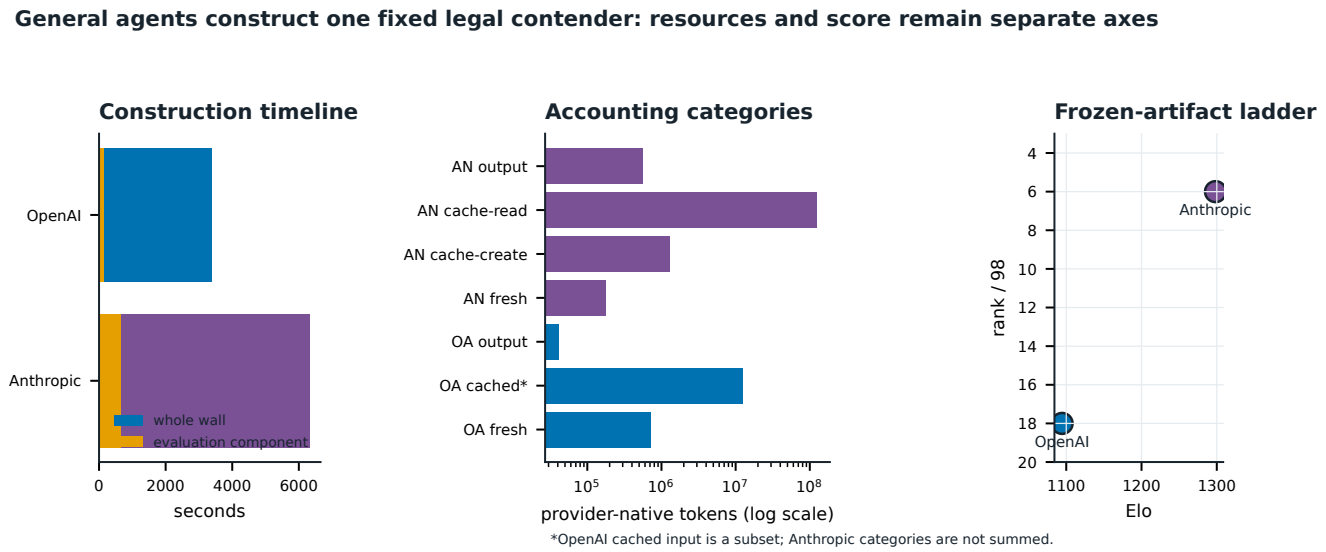

Fig. 5. Agent challenge. Left: evaluation is contained within construction wall time. Center: provider-native categories are displayed without a synthetic cross-provider total. Right: frozen-artifact standings.

TABLE IV
MOST COMMON OPERATOR MOTIFS OCCUR AMONG WINS AND LOSSES.

Operator	wins	losses	within-family win %
pair_prod	128	80	61.5
sigmoid	88	71	55.3
winsor_clip	71	72	49.7
none	85	46	64.9
rand_proj	73	51	58.9
cheb	80	35	69.6
standardize	64	49	56.6
robust_scale	60	43	58.3
quantile_bins	41	38	51.9
blom	40	38	51.3

TABLE V
EXPLICIT NEGATIVE RESULTS: THE LARGEST, TRAIN-CONFIDENT, AND SLOW LOSSES.

Type	case hash	robust	margin
largest loss	`84e307f5c421b511`	0.199	-0.025
largest loss	`49eb9093b61a7555`	0.276	-0.024
largest loss	`1674565826a12d26`	0.272	-0.024
confident loss	`92b8caea90d27b46`	1.000	-0.016
confident loss	`5b3e8ece3ea63fc5`	0.998	-0.023
confident loss	`55f825e2764bfdf4`	0.995	-0.016
slow loss	`5b3e8ece3ea63fc5`	0.998	-0.023
slow loss	`b8520469902552ee`	0.250	-0.010
slow loss	`e7c38c954a3bd6a5`	0.544	-0.024

concentration is shown rather than treated as independent replication. Descriptive source-group resampling, if used, is a stability diagnostic only.

B. Matched win/loss contrasts

For each focal win, the loss twin is frozen by rule before explanatory prose: require a different source group; prefer the same head; minimize Euclidean distance in standardized log train rows and log feature count; break ties by operator-family Jaccard distance and dataset hash. These comparisons are post-hoc and non-causal. The selected loss hashes are `f7b06385de3bc742`, `f7e6abfacdac77c4`, and `da128a8ca7a63ac2`, with signed margins -0.0031, -0.0100, and -0.0222. A shared head or primitive therefore cannot itself explain success.

VIII. THREE POST-HOC MECHANISM STUDIES

The evidence-card format is fixed: source and split hashes, legal genome, plain-language mechanism, train-side sensitivity, all four foundation reports, three clocks, leakage and duplicate checks, a cross-source loss twin, and a strict claim boundary. Table VI gives the held-out values. The comparator labeled strongest is always the minimum foundation RMSE, never the largest factor.

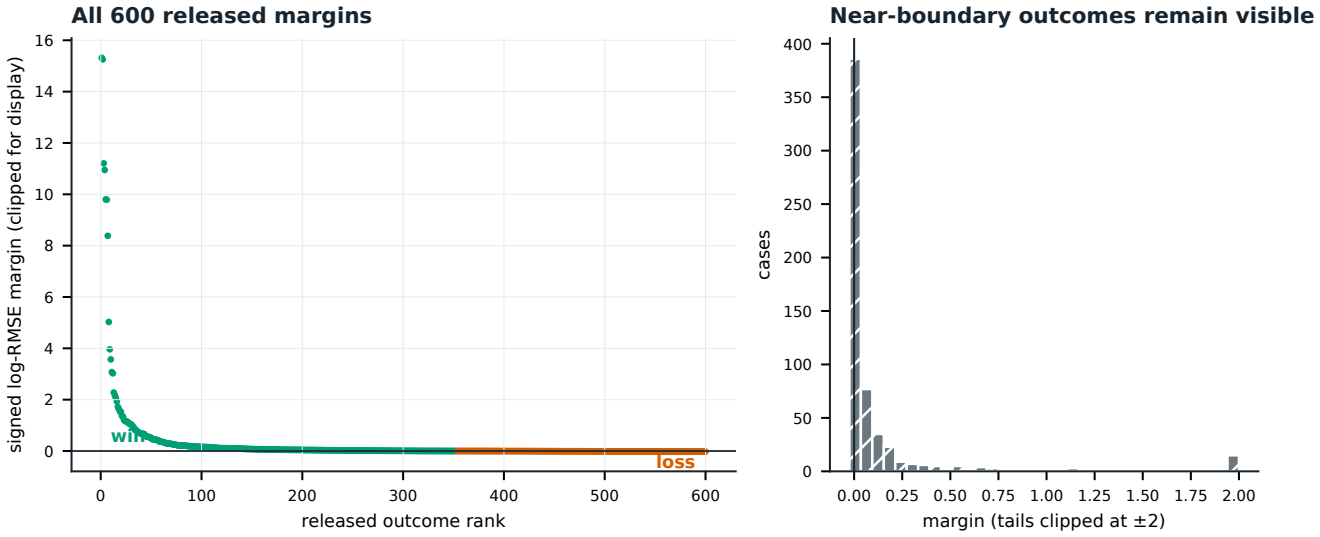

Fig. 6. All 600 released margins. Rank used the observed RMSE ratio; tails are clipped only for display. The zero boundary and near-zero cases remain visible. No population inference is attached to the outcome-ranked corpus.

TABLE VI
CASE EVIDENCE FROM THE OUTCOME-RANKED ATLAS. "STRONGEST FOUNDATION" MEANS THE MINIMUM RMSE AMONG THE FOUR STORED REPORTS.

Case	Train	Test	p	IZ RMSE	strongest foundation	factor
Spectrometer	442	89	101	2.706836e-06	0.0118394	4,373.89×
Greenhouse gases	250	50	2	0.005004426	0.0170191	3.40×
CPU performance	174	35	7	0.002653884	0.0136234	5.13×

A. *Spectrometer: simple calibration*

The split contains 442 training rows, 89 test rows, and 101 spectral features. The legal artifact is ridge-GCV with no added transform. Its RMSE is 2.706836e-06; the strongest recorded foundation RMSE is 0.0118394, a 4,373.89-fold difference. Search took 20.440 s, fit/predict 0.563 s, and the recorded workflow 100.180 s.

The source/split audit confirms target-last training layout, target absence from the test table, no feature identical to the target, no exact duplicate feature pair, and bound train/test hashes. The standardized spectral matrix has effective rank 2.83; 30 singular directions account for the declared high-energy threshold. Ridge learning curves and cross-fold coefficient cosine stability support, but do not prove, a low-dimensional calibration interpretation. We therefore state only that the result is *consistent with a low-dimensional calibration structure*. The matched loss shows that ridge-family simplicity is not universally sufficient.

B. *Greenhouse gases: explicit temporal structure*

This case contains 250 training rows, 50 test rows, and 2 encoded features (year and gas). The selected artifact is GPR with an RBF–periodic composition. RMSE is 0.005004426 versus strongest foundation RMSE 0.0170191, a 3.40-fold difference. The three clocks are 15.955 s search, 1.224 s fit/predict, and 19.614 s workflow.

Train-side arms compare the selected kernel family with RBF-only, periodic-only, linear, and mean baselines. A separate sensitivity treats the gas identifier categorically instead of assigning metric geometry to its numeric encoding. Because the stock multidimensional periodic distance is numerically delicate, that sensitivity uses a fixed declared diagonal term rather than outcome tuning. The case demonstrates an interpretable inductive-bias match on one encoded table, not a general climate-modeling claim.

C. *CPU performance: constructed interactions*

The CPU table has 174 training rows, 35 test rows, and 7 hardware features. Its genome repeatedly constructs bounded polynomial interactions and sigmoid features, then applies ridge. RMSE is 0.002653884 against strongest foundation RMSE 0.0136234, a 5.13-fold difference. Search took 17.578 s, fit/predict 0.480 s, and the recorded workflow 19.527 s.

In plain language, the genome builds a compact nonlinear basis while ridge stabilizes small-sample fitting. Train-side arms remove interactions, remove sigmoids, and remove ridge regularization; a vendor sensitivity contrasts numeric and one-hot treatment. The held-out split is small, so uncertainty language relies on train-side fold dispersion and the fixed reported score, not repeated test-based model selection. The loss twin demonstrates that similar feature construction can be unstable or mismatched.

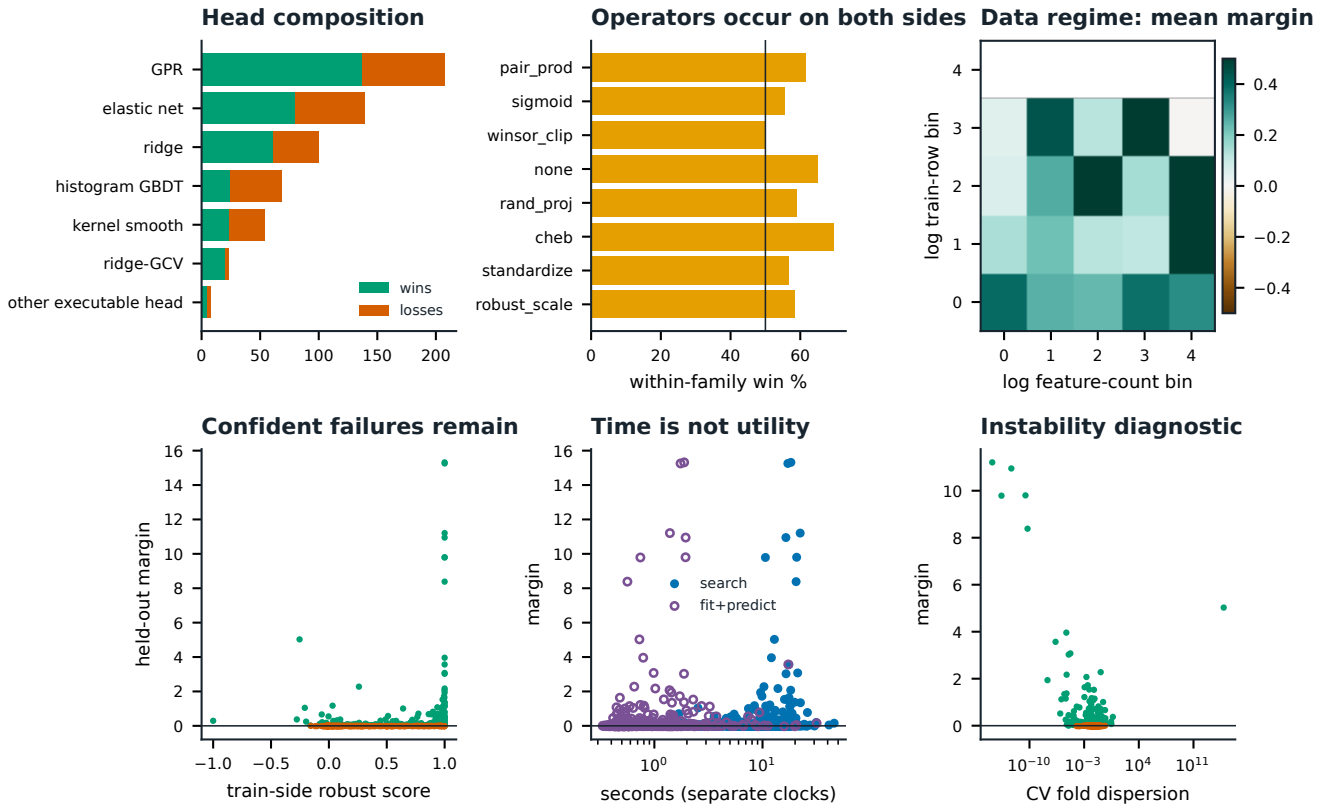

Fig. 7. Descriptive atlas diagnostics. Counts retain all 353 wins and 247 losses. Head/operator panels are composition; regime, robust-score, timing, and fold-dispersion panels are post-hoc associations. Search and fit/predict clocks are separate.

TABLE VII
SELECTED TRAIN-SIDE MECHANISM ARMS. THESE ARE POST-HOC SENSITIVITY CHECKS, NOT CAUSAL ESTIMATES.

Case	train-side arm	mean RMSE
Spectrometer	mean	0.155014
Spectrometer	ridge (inner CV)	2.89569e-06
Greenhouse	RBF $imes$ periodic	0.0470105
Greenhouse	RBF only	0.0140886
Greenhouse	periodic only	0.125124
CPU	base ridge	0.0467157
CPU	interactions + sigmoid + ridge	0.00518405
CPU	interactions without ridge	0.00541586

Appendix C reports all four foundation comparisons rather than only the strongest row.

IX. Closest Work and Novelty Boundary

Classical genetic programming searches executable structures and develops reusable representations of program space [1], [2], [3], [4]. Neural program synthesis learns to generate programs from examples or specifications [5], [6], [7], [8]. Symbolic regression and equation discovery search interpretable expressions [9], [10], [11], [12], [13]. These lines establish that program structure can be searched and learned; they do not by themselves test which complete learning program should be proposed first for a new table.

AutoML configures or selects models under a task-time search budget [14], [15], [16], [17], [18], [19]. AutoML-Zero evolves learning algorithms from primitive operations [20]. Meta-learning learns adaptation rules or optimizers [21], [22], [23], [24]. Our novelty delta is narrower: learn from independently scored program histories to propose one legal, standalone, dataset-conditioned algorithm, and test the $N = 1$ boundary separately from task-time selection.

Prior-data fitted networks and tabular foundation models learn reusable predictive priors [25], [26], [27], [28], [29], [30]. Strong trees and neural tabular learners supply the surrounding predictive landscape [31], [32], [33], [34], [35], [36], [37], [38], [39], [40]. Those systems primarily learn a predictor or fixed inference procedure; InnovationZero learns which executable program to propose.

Verifier-driven discovery is the closest conceptual family. FunSearch couples generated programs to an evaluator [41]; AlphaTensor and AlphaDev frame algorithm discovery as measured games [42], [43]. Broader scientific systems demonstrate prediction, structure discovery, or tool-mediated research [44], [45], [46]. Self-play and temporal-difference systems show how experience can improve action selection [47], [48], [49], [50],

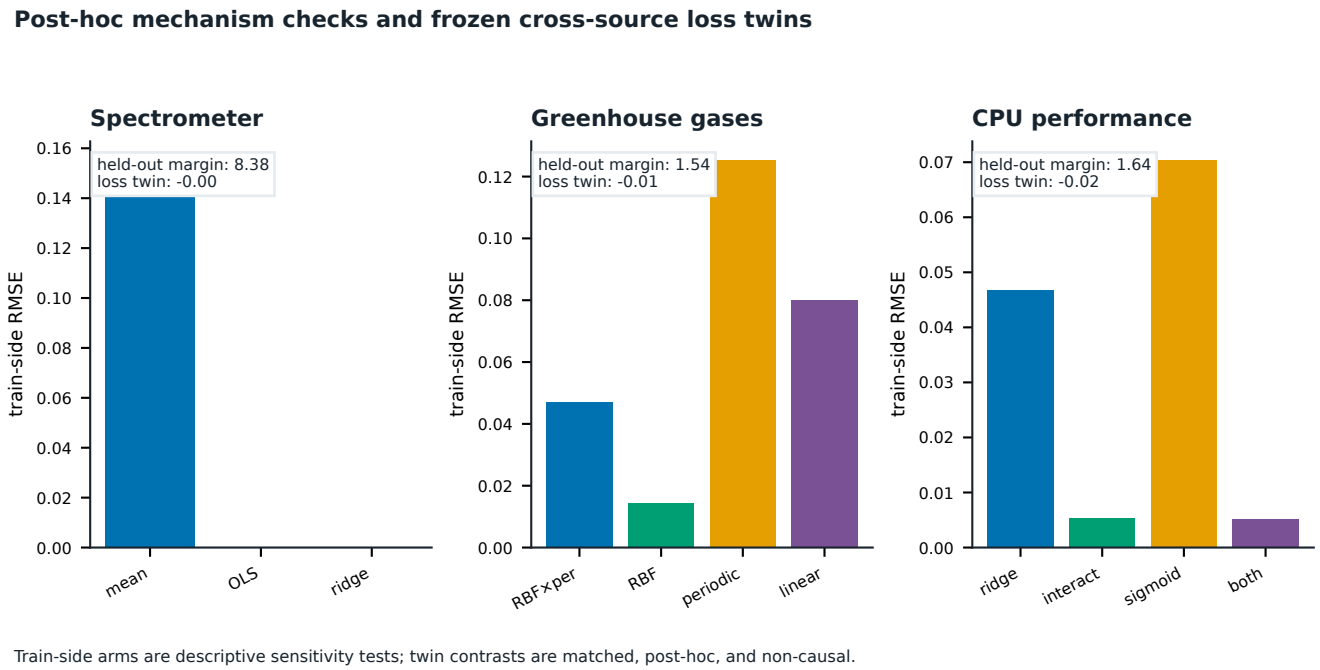

Fig. 8. Train-side sensitivity arms and fixed cross-source loss twins. Bars do not select the held-out artifact; twin contrasts are descriptive and non-causal. Case search, fit/predict, and workflow clocks remain separately receipted in the evidence cards.

[51], [52], [53]; unlike those policy-gradient or value-learning examples, the present neural update uses supervised sequence and ranking surrogates over external receipts. Statistical references for descriptive stability checks are Wilson and bootstrap methods [54], [55], but we deliberately attach no population interval to the outcome-ranked atlas.

The claim is therefore neither "first program search" nor "new mathematics." It is the conjunction of a dataset-to-program foundation model, a legal local artifact boundary, externally measured feedback compiled into a first-proposal policy, and explicit separation of strict, bounded, frontier, agent, and atlas evidence.

X. Limitations and Failure Modes

First, the atlas was selected by observed outcome. It can reveal motifs, counterexamples, and mechanisms worth testing; it cannot estimate prospective frequency. Second, case studies are post-hoc and cannot establish causality. Third, grammar bounds make the game auditable but restrict the space of possible algorithms. Fourth, the strict field result is tied to one checkpoint, field, dataset protocol, and scorer. Fifth, a structurally new genome may recombine familiar primitives; utility does not prove mathematical novelty.

Sixth, source groups repeat within the atlas and are displayed as concentration rather than independent replication. Seventh, a confident train-side score can still fail held-out, which motivates the negative table and conditional ablation. Eighth, timings are platform receipts rather than hardware-normalized complexity. Ninth, external-agent token accounting is provider-specific. Finally, the live GPT-5.6 review campaign is a process audit, not scientific evidence, and cannot contribute a score until all identity and artifact-binding gates pass.

XI. Conclusion

InnovationZero operationalizes a data-to-code foundation model: represent the problem, emit one legal genome, compile standalone code, measure it independently, and train the next proposal from receipted surrogate targets. The strict first move is the central result; bounded and frontier scores remain separate diagnostics. The released 600-case atlas supplies both successes and failures, not a prospective win-rate claim. Together, the field result, executable contract, gradient boundary, general-agent receipts, and matched mechanism studies support a calibrated thesis: repeated verified program construction can be amortized into a policy that proposes useful executable algorithms, while success still depends on matching inductive bias to data.

Appendix A
Architecture and Compatibility Receipt

TABLE VIII
Checkpoint details moved out of the conceptual architecture.

Component	receipted value
parameters	27,208,732
dataset embedding	1,536
target summary	512
model width	512
decoder layers	6
attention heads	4
feed-forward width	2,048
vocabulary	1,103
maximum sequence	128
reward heads	4
optional auxiliary classes	14

The context encoder combines a dataset representation with a target summary and projects them into the decoder width. A

causal Transformer emits the grammar sequence. Reward-token embeddings are distinct from decoder token embeddings; their pooled genome representation interacts with dataset context before the bootstrap reward heads. The genome-only control consumes a detached representation. Runtime and instability are dataset-conditioned auxiliary heads.

Compatibility loading never creates labels. Older corpus rows may omit rank, runtime, instability, or auxiliary family fields; each objective uses an explicit availability mask. Checkpoint loading reports missing and unexpected parameters. Optional fields stay inactive until evidenced.

Appendix B
Loss Construction and Collapse Audit

Each training row can contribute to different objectives. Legal but unscored rows contribute sequence imitation. Rows with finite within-dataset scores contribute pairwise and list-wise ranking. Rows with runtime train the log-Huber head. Evaluation errors can train instability without entering the finite ranking order. This partition prevents missingness from becoming an outcome.

Bootstrap membership is deterministic from dataset, engine, genome identity, and head index. The conditional reward mean is used for ranking, while standard deviation across heads is an epistemic disagreement diagnostic. The genome-only ablation is validated separately: if it matches the conditional model, evidence for dataset conditioning has collapsed. Ranking remains active throughout training and is not uncertainty-downweighted. Log-variance clamps prevent a learned task weight from diverging. Former-self margins prevent mere ties with a retained ancestor from appearing as improvement.

This is not policy-gradient reinforcement learning. The evaluator does not return a differentiable trajectory. It writes a score receipt that later becomes supervised/ranking data.

Appendix C
Case Evidence Cards

Each card binds the source dataset hash, train/test file hashes, genome and genome hash, target-column audit, duplicate-feature audit, train-side partition seed, all four comparison receipts, and three clocks. Raw train/test tables, predictions, and residual arrays are excluded from the release. The derived mechanism summary is reproducible with the source-root command and contains no row-level data.

The spectrometer card tests spectral decay, effective rank, learning curves, coefficient stability, and nested-CV ridge comparison. The greenhouse card tests kernel families and categorical gas encoding. The CPU card tests interaction, sigmoid, regularization, and vendor-encoding arms. None changes the frozen held-out artifact.

Appendix D
Reproducibility and Security Contract

The offline command is:

```
python3 reproduce.py -verify
```

A quick rebuild regenerates atlas analyses, claims, tables, vector figures, and PDF from packaged evidence. A full rebuild additionally accepts the ground-truth source root and reruns the three mechanism summaries. JSON is canonicalized; archive members are sorted; timestamps, uid/gid, names, and permissions are normalized; gzip metadata is fixed; symlinks, devices, absolute paths, and traversal are rejected.

Only two derived CSV summaries are allowed: the 600-row case index and the focal-case table. Their headers, row counts, identities, and hashes are verified. Raw train/test tables, predictions, residuals, checkpoints, dense arrays, browser state, credentials, and secret patterns are rejected. Every plotted datum is copied from one figure-data JSON whose hash is bound in the figure manifest.

Appendix E
Protocol Misstatements That Fail Review

The following substitutions are invalid: calling the atlas blind; calling its composition a prospective win rate; replacing strict first move with the frontier maximum; calling external receipt training differentiable RMSE or policy-gradient RL; turning missing reports into losses; summing incompatible Anthropic token categories; admitting GPT-5.6 performance without a bound receipt; or presenting a post-hoc case as a causal mechanism or new mathematics.

Appendix F
Canonicalization, Compilation, and Failure Semantics

A. Parser state and canonical identity

The decoder state is a parser state, not merely a token position. It records whether the pipeline has opened, which transform parameters remain, whether a terminal head has been emitted, current symbolic depth, node count, feature-width budget, and total sequence budget. Legal-next-token masking is computed from this state before probability normalization. Once a head is emitted, only closure and termination tokens remain legal. A token sequence that reaches the length cap before syntactic closure is a failed proposal rather than an implicitly repaired program.

Canonicalization parses surface syntax into typed operations, normalizes parameter order and numeric representation, removes packaging-only aliases, and serializes one stable genome. Identity includes the compiler engine and canonical genome. Two differently formatted strings with the same typed operations are one action; two pipelines that share a head but differ in transforms remain different actions. The genome hash therefore binds semantics at the declared compiler version rather than typography.

This identity rule is also the deduplication boundary for experience. It prevents a frequently formatted genome from receiving extra imitation mass and prevents packaging aliases from masquerading as independent discoveries. Historical family labels remain provenance, not repeated statistical units.

B. Fold-local state

Every operator is classified as stateless or fitted. Stateless arithmetic is applied directly after deterministic sanitation.

TABLE IX
ALL FOUR RECORDED FOUNDATION COMPARISONS FOR EACH FOCAL CASE. FACTOR IS FOUNDATION RMSE DIVIDED BY THE FIXED INNOVATIONZERO RMSE.

Case	foundation report	RMSE	factor
Spectrometer	TabFM Ensemble	0.01183941	4373.89×
Spectrometer	TabFM	0.0147104	5434.54×
Spectrometer	TabPFN-v3	0.01226374	4530.65×
Spectrometer	TabPFN-v3 Default	0.01224753	4524.67×
Greenhouse gases	TabFM Ensemble	0.02332562	4.66×
Greenhouse gases	TabFM	0.01701912	3.40×
Greenhouse gases	TabPFN-v3	0.02516621	5.03×
Greenhouse gases	TabPFN-v3 Default	0.0252868	5.05×
CPU performance	TabFM Ensemble	0.01362337	5.13×
CPU performance	TabFM	0.01767331	6.66×
CPU performance	TabPFN-v3	0.03008721	11.34×
CPU performance	TabPFN-v3 Default	0.02972185	11.20×

Scaling, projection, quantile, binning, target-aware encoding, kernel parameter, and model-head state is learned from the training fold only. Validation rows are transformed with the fitted state. The final executable repeats the same order after fitting once on the supplied training table. A primitive is not admissible merely because the parser recognizes it: train-side runtime, emitted runtime, and validation semantics must agree.

The compiler never obtains test labels. Reading test feature rows is limited to deterministic transformation and prediction after training state is frozen. Transductive fitting on test covariates is outside this contract even when labels are absent, because it changes the declared algorithm and complicates comparison with inductive baselines.

C. Failure taxonomy

Failures are typed. Syntax failure means the token stream cannot form a legal pipeline. Canonicalization failure means parameters or expression bounds are invalid. Compile failure means the legal specification lacks a supported local implementation. Fit failure includes numerical exceptions and resource limits. Prediction failure includes wrong length, nonfinite values, and unstable ordering. Coverage failure means the evaluator has no valid report. Only a finite prediction vector reaches RMSE and pairwise scoring. The receipt retains failure type, stage, deterministic seed, and elapsed clock; it does not invent a replacement score.

This distinction matters for learning. A failed row can train legality or stability, and may still contribute syntax imitation if its target genome is known to be legal in another runtime. It cannot enter Bradley–Terry or listwise order as an arbitrary worst finite value. Missing coverage is still less informative and contributes no outcome loss.

APPENDIX G
ATLAS DERIVATIONS AND SENSITIVITY RULES

The atlas compiler reads only the released case receipts and asserts a continuous rank sequence, unique dataset hashes, the locked outcome composition, finite nonnegative RMSE values, and equality between stored and recomputed ratios. It then writes an analysis JSON and a smaller figure-data JSON. Their hashes are linked by a build receipt. Plotting code is prohibited from loading case JSONs directly, which gives each visual one auditable data boundary.

Head families are mutually exclusive because every genome ends in one head. Operator families are multi-label: a case contributes once to each distinct operator family in its genome. Consequently, operator counts cannot be summed to the atlas denominator. Each motif row reports raw wins, raw losses, its share of atlas cases, and win percentage within the motif. A high within-motif percentage remains post-hoc because the atlas was outcome-ranked and motifs were not randomized.

The regime heatmap uses fixed bins in log training rows and log feature count. Empty cells remain missing rather than zero-margin. Timing diagnostics preserve search, finalist evaluation, fit/predict, and total workflow fields; only search and fit/predict are plotted together, with distinct markers. Robust-score and fold-dispersion panels retain negative outcomes so confident and stable-looking failures remain visible.

Denominator sensitivity is a filter, not a new ranking. For each declared floor, cases whose TabFM Ensemble RMSE falls at or below the floor are removed from the already frozen order. The remaining cases are not replenished from an unreleased cohort. This prevents a sensitivity analysis from silently creating a different selected sample.

Source-group concentration uses the stored connected-component label. It reports unique groups, repeated groups, largest group, and the most concentrated groups. A source-group bootstrap may describe stability of an atlas statistic, but cannot repair outcome selection or supply a population confidence interval.

The twin matcher standardizes log row and feature counts once over all released cases. For each focal win it removes losses from the same source group, restricts to the same head when at least one such loss exists, minimizes size distance, and uses operator Jaccard distance only as a tie-breaker before dataset hash. Explanatory text is written after these identities are frozen. This produces a reproducible contrast, not a causal match.

APPENDIX H
TEN-EPOCH REVIEW CAMPAIGN BOUNDARY

The review dossier defines ten sequential epochs, seven fresh conversations per epoch, and a visible exact-model gate

for GPT-5.6. Candidate revisions are immutable: epoch e01 receives r00, and each sealed repair can create only the next revision. Reviewers receive the current candidate and stable rubric, not prior scores, reviews, or repair plans during fresh scoring. Each conversation has one specialist emphasis but scores the entire rubric.

Model proof requires a visible ChatGPT model control before submission and after completion. The receipt binds a pseudonymous profile, hashed conversation URL, visible selector text, redacted DOM hash, screenshot hash, and timestamps. A URL parameter, configuration string, self-report, or event-log echo is insufficient. Any missing or mismatched tab receipt stops the epoch with the declared model-identity exit code; no alternate GPT model, provider, or local mock can count.

Freshness is checked across the complete campaign by conversation URL hashes and tab identities. Every epoch must contain exactly seven Markdown/JSON pairs conforming to the v2 review schema. Review JSON supplies category scores A–J, an overall score, cited findings, files examined, and the model receipt reference. Exact semantic duplicates may be merged only after all seven reviews are sealed; minority objections remain in the finding ledger.

Category medians form the uncapped score and reviewer totals supply the interquartile range. An evidenced automatic blocker caps the score. Submission readiness additionally requires the threshold, no blocker, no unresolved major, and no category below its fractional floor. The final campaign gate also checks late-epoch scores, robust trend, complete pre/post identity receipts, and that the final fresh score is the campaign maximum. The trajectory is never forced to be monotone; regressions remain evidence of a failed repair.

Repairs are performed outside reviewer conversations. A repair receipt maps every finding to fixed, accepted limitation, rebutted with evidence, or open; records before/after hashes and changed paths; declares whether evidence-derived artifacts changed; and stores verification results. Sealing appends one immutable history point and advances the campaign state. The final red-team archive excludes browser state and credentials. Until all live gates pass, the campaign status remains not submission-ready and no review score is represented as scientific evidence.

Appendix I
Claim Ledger and Evidence Separation Audit

The claim ledger is generated rather than curated after typesetting. Each macro record contains its TeX name, rendered value, source file, and JSON pointer. Every numeric table row records its source object. A build receipt hashes the ledger, claims file, generated tables, and focal-case CSV. Manuscript verification then scans the source, abstract, captions, tables, and appendices for excluded cohort literals and stale macro names. This is a guard against a scientifically invalid but easy editorial regression: reintroducing a broader denominator while revising prose about the released atlas.

Evidence is divided into four namespaces. *Sealed-field evidence* contains strict, bounded, and frontier receipts and is indexed by evaluation mode. *Agent evidence* binds a construction session to one frozen artifact and its challenge score. *Atlas evidence* contains only released case identities and post-hoc derived analyses. *Process evidence* contains build, archive, and red-team receipts. A value cannot move between namespaces merely because its unit matches: field Elo is not agent Elo, atlas margin is not train-side robust score, and a reviewer score is not model performance.

Case identity has three layers. The dataset layer binds source and split hashes. The artifact layer binds canonical genome, implementation, and checkpoint. The outcome layer binds candidate, comparator, RMSEs, factor, and selection rank. Recomputing a factor checks the outcome layer but cannot prove the dataset or artifact layer; all three must be present for a publishable case statement. The strongest-foundation assertion adds an executable minimum-over-four check so a larger but rhetorically convenient factor cannot replace the best comparator.

Clock identity is similarly explicit. Search time measures proposal work, finalist time measures additional train-side comparison where recorded, fit/predict measures execution of the frozen artifact, and workflow time measures the enclosing recorded lifecycle. Component clocks may overlap or be contained in the whole. The verifier checks identities but does not infer additivity. Agent evaluation is likewise a component of construction wall time.

Token identity preserves provider semantics. OpenAI cached input is a subset of input and reasoning output is a subset of output, permitting its receipted arithmetic identity. Anthropic reports fresh input, cache creation, cache read, and output categories separately; no derived total field exists in the evidence schema or generated macros. Emitted source-code length is stored outside both providers' usage categories.

Citation verification has two passes. The structural pass requires every cited key to resolve and every bibliography entry to be used. The context pass groups claims into program synthesis, genetic programming, AutoML/meta-learning, prior-fitted and tabular foundation models, verifier-driven discovery, and equation discovery. Each novelty statement is checked against the primary work's actual output object: predictor, selected configuration, synthesized program, or learned proposal. Title, authors, venue, year, DOI, URL, and version are retained in the citation ledger so a correct-looking key cannot mask mismatched content.

Finally, deterministic reproduction is checked twice: two builds in the source tree must match, and a build from clean extraction must reproduce the generated evidence, figures, PDF, manifest, and archive. The clean extractor rejects links, devices, traversal, and absolute paths before writing. This does more than protect the archive; it tests that no undeclared local path, browser state, model cache, or private source tree is required for the published offline verification.

References

[1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992. [Online]. Available: https://mitpress.mit.edu/9780262111706/genetic-programming/

[2] W. B. Langdon and R. Poli, *Foundations of Genetic Programming*. Springer, 2002. [Online]. Available: https://doi.org/10.1007/978-3-662-04738-7

[3] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Lulu, 2008. [Online]. Available: http://www.gp-field-guide.org.uk/

[4] J. Schmidhuber, "Evolutionary principles in self-referential learning," Ph.D. dissertation, Technische Universität München, 1987. [Online]. Available: https://people.idsia.ch/~juergen/firstpow.html

[5] S. Gulwani, O. Polozov, and R. Singh, "Program synthesis," in *Foundations and Trends in Programming Languages*, vol. 4, no. 1–2, 2017, pp. 1–119. [Online]. Available: https://doi.org/10.1561/2500000010

[6] M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow, "Deepcoder: Learning to write programs," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=ByldLrqlx

[7] J. Devlin, J. Uesato, S. Bhupatiraju, R. Singh, P. Mirowski, and P. Kohli, "Robustfill: Neural program learning under noisy i/o," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 990–998. [Online]. Available: https://proceedings.mlr.press/v70/devlin17a.html

[8] K. Ellis, C. Wong, M. Nye, M. Sablé-Meyer, L. Morales, L. Hewitt, L. Cary, A. Solar-Lezama, and J. B. Tenenbaum, "Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning," in *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021, pp. 835–850. [Online]. Available: https://doi.org/10.1145/3453483.3454080

[9] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009. [Online]. Available: https://doi.org/10.1126/science.1165897

[10] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, p. eaay2631, 2020. [Online]. Available: https://doi.org/10.1126/sciadv.aay2631

[11] M. Cranmer, "Interpretable machine learning for science with PySR and symbolicregression.jl," *arXiv preprint arXiv:2305.01582*, 2023. [Online]. Available: https://arxiv.org/abs/2305.01582

[12] B. K. Petersen, M. Landajuela, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and B. Arnold, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=m5Qsh0kBQG

[13] C. Cornelio, S. Dash, V. Austel, T. R. Josephson, J. Goncalves, K. L. Clarkson, and N. Megiddo, "Combining data and theory for derivable scientific discovery with AI-Descartes," *Nature Communications*, vol. 14, p. 1777, 2023. [Online]. Available: https://doi.org/10.1038/s41467-023-37236-y

[14] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. Murphy, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*, 2011, pp. 507–523. [Online]. Available: https://doi.org/10.1007/978-3-642-25566-3_40

[15] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD*, 2013, pp. 847–855. [Online]. Available: https://doi.org/10.1145/2487575.2487629

[16] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, vol. 28, 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html

[17] R. S. Olson and J. H. Moore, "Tpot: A tree-based pipeline optimization tool for automating machine learning," in *Proceedings of the Workshop on Automatic Machine Learning*, 2016, pp. 66–74. [Online]. Available: https://proceedings.mlr.press/v64/olson_tpot_2016.html

[18] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5

[19] J. Wang *et al.*, "Automated machine learning: A survey," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021. [Online]. Available: https://doi.org/10.1016/j.knosys.2020.106622

[20] E. Real, C. Liang, D. So, and Q. V. Le, "AutoML-Zero: Evolving machine learning algorithms from scratch," *Proceedings of Machine Learning Research*, vol. 119, pp. 8007–8019, 2020. [Online]. Available: https://proceedings.mlr.press/v119/real20a.html

[21] J. Vanschoren, "Meta-learning: A survey," in *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019, pp. 35–61. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5_2

[22] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 1126–1135. [Online]. Available: https://proceedings.mlr.press/v70/finn17a.html

[23] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rJY0-Kcll

[24] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, 2022. [Online]. Available: https://doi.org/10.1109/TPAMI.2021.3079209

[25] S. Müller, N. Hollmann, S. P. Arango, J. Grabocka, and F. Hutter, "Transformers can do bayesian inference," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=KSugKcbNf9

[26] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter, "TabPFN: A transformer that solves small tabular classification problems in a second," *arXiv preprint arXiv:2207.01848*, 2023. [Online]. Available: https://arxiv.org/abs/2207.01848

[27] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter, "Accurate predictions on small data with a tabular foundation model," *Nature*, vol. 637, pp. 319–326, 2025. [Online]. Available: https://doi.org/10.1038/s41586-024-08328-6

[28] L. Grinsztajn, K. Flöge, O. Key, F. Birkel, P. Jund, B. Roof, M. Manium, S. B. Hoo, M. Bühler, A. Garg *et al.*, "TabPFN-3: Technical report," *arXiv preprint arXiv:2605.13986*, 2026. [Online]. Available: https://arxiv.org/abs/2605.13986

[29] W. Kong and A. Das, "Introducing TabFM: A zero-shot foundation model for tabular data," Google Research, 2026. [Online]. Available: https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/

[30] N. Erickson, L. Purucker, A. Tschalzev, D. Holzüller, P. M. Desai, D. Salinas, and F. Hutter, "TabArena: A living benchmark for machine learning on tabular data," *arXiv preprint arXiv:2506.16791*, 2025. [Online]. Available: https://arxiv.org/abs/2506.16791

[31] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[32] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD*, 2016, pp. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[33] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: https://doi.org/10.1214/aos/1013203451

[34] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 6679–6687. [Online]. Available: https://doi.org/10.1609/aaai.v35i8.16826

[35] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein, "SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training," *arXiv preprint arXiv:2106.01342*, 2021. [Online]. Available: https://arxiv.org/abs/2106.01342

[36] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 18 932–18 943. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html

[37] Y. Gorishniy, I. Rubachev, N. Kartashev, D. Shlenskii, A. Kotelnikov, and A. Babenko, "TabR: Tabular deep learning meets nearest neighbors," in *International Conference on Learning Representations*, 2024. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/file/4ef594af0d9a519db8fb292452c461fa-Paper-Conference.pdf

[38] S. Popov, S. Morozov, and A. Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1eIiCNYwS

[39] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, "Tabtransformer: Tabular data modeling using contextual embeddings," *arXiv preprint arXiv:2012.06678*, 2020. [Online]. Available: https://arxiv.org/abs/2012.06678

[40] G. Ke, Z. Xu, J. Zhang *et al.*, "Deepgbm: A deep learning framework distilled by gradient boosting machines," *arXiv preprint arXiv:1802.06930*, 2019. [Online]. Available: https://arxiv.org/abs/1802.06930

[41] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. R. Ruiz *et al.*, "Mathematical discoveries from program search with large language models," *Nature*, vol. 625, pp. 468–475, 2024. [Online]. Available: https://doi.org/10.1038/s41586-023-06924-6

[42] A. Fawzi, M. Balog, A. A. Horváth *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, pp. 47–53, 2022. [Online]. Available: https://doi.org/10.1038/s41586-022-05172-4

[43] D. J. Mankowitz, G. Michalewski, A. Bezerra *et al.*, "Discovering faster sorting algorithms using deep reinforcement learning," *Nature*, vol. 618, pp. 257–263, 2023. [Online]. Available: https://doi.org/10.1038/s41586-023-06004-9

[44] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03819-2

[45] A. Merchant, S. Batzner, S. S. Schoenholz *et al.*, "Scaling deep learning for materials discovery," *Nature*, vol. 624, pp. 80–85, 2023. [Online]. Available: https://doi.org/10.1038/s41586-023-06735-9

[46] D. Boiko, R. MacKnight, and G. Gomes, "Autonomous chemical research with large language models," *Nature*, vol. 624, pp. 570–578, 2023. [Online]. Available: https://doi.org/10.1038/s41586-023-06792-0

[47] G. Tesauro, "Temporal difference learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995. [Online]. Available: https://doi.org/10.1145/203330.203343

[48] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988. [Online]. Available: https://doi.org/10.1007/BF00115009

[49] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[50] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016. [Online]. Available: https://doi.org/10.1038/nature16961

[51] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[52] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. [Online]. Available: https://doi.org/10.1126/science.aar6404

[53] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, pp. 604–609, 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-03051-4

[54] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927. [Online]. Available: https://doi.org/10.1080/01621459.1927.10502953

[55] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979. [Online]. Available: https://doi.org/10.1214/aos/1176344552