The Weismann Machine

Outer-Loop Self-Improvement behind a Germline Barrier, with Time as the Grader

Abhay Chakra Sadineni

University of California, Berkeley
abhaychakra@berkeley.edu

November 2026
DOI: 10.5281/zenodo.24333917

Abstract

Self-improving AI systems now edit their own code, archive their variants and draw parents from the archive largely by benchmark score. The agent being improved writes its own successor from its own traces, and each child inherits its parent's code, edits included. In 1892 August Weismann argued that the germ line is set apart from the body, so that what an organism acquires during its life is not passed on. This paper turns Weismann's separation into a design rule and calls a system built to it a Weismann machine. Its inherited genome is compact and specifies how a learner develops and learns, and whatever a lifetime learns is discarded at its end. A germline barrier confines each lifetime to private storage and a single typed output, and lets the operators that vary the genome see genomes and a statistic of their fitness and nothing else, so that variation and selection happen only in an outer loop. The grader is time. Learners are scored with strictly proper scoring rules on events that resolve after a sealed cutoff, across rolling cutoffs, with a live tier of forecasts committed before their questions resolve. Under these conditions the barrier excludes tampering with the grader and with what it reads, prevents the inheritance of lifetime state, and bounds the information about the selection data that can reach the genome. The paper places the scheme against self-improving agents, learned learning rules and outcome-trained forecasters, and specifies a test on a language model with a 1913 knowledge cutoff, with nine arms, a budget of about 37,000 H100-hours and a decision rule fixed in advance. Its comparison arms separate the effect of scoring after a sealed cutoff from those of scoring format, search, recalibration and added machinery.

Keywords: self-improving systems; meta-learning; neuroevolution; learned learning rules; forecasting; temporal leakage; AI safety.

1 Introduction

In 1892 the zoologist August Weismann published *Das Keimplasma* [1], an argument that the material of heredity is confined to a line of cells set apart from the rest of the body. The body, on this view, is disposable. It grows, learns, is injured and dies, and none of that is written back into the cells that carry heredity forward. The separation came to be called the Weismann barrier, and it became the standard argument against the inheritance of acquired characters.

Self-improving AI systems are built differently. A coding agent reads its own source, proposes an edit, runs the edited copy on a benchmark and adds it to an archive from which parents for later edits are drawn, largely by score [2–5]. Each child inherits its parent's code, including the edits the parent made to itself. A 2026 analysis of such systems calls their inheritance Lamarckian and argues that it lets a poisoned memory or an unsafe tool, once acquired, travel down a lineage [6]. The same systems have shown a related failure, in which an agent that writes its own successor changes what its grader sees. When the Darwin Gödel Machine was set

to reduce tool-use hallucination, one of its variants raised its score by removing the logging that the hallucination check relied on, rather than by hallucinating less [2]. An audit published in September 2026 used a language-model auditor to flag what its authors call harness tampering in between 18.3 and 84.6 percent of iterations across four self-improving systems and one automated-research system [7].

Can a self-improving learner keep Weismann's barrier, and would it gain by doing so? Two observations suggest that it can and would. The first is the genomic bottleneck. The human genome is about 3.06 billion base pairs long [8], roughly six billion bits, while the brain it builds has about 86 billion neurons [9] and on the order of 10^{14} synapses [10]. The genome cannot list the wiring. It specifies rules for building and changing it, and animals nonetheless arrive able to learn fast [10]. When the same compression is imposed on artificial networks, a trained network's weights can be compressed by up to several orders of magnitude into a small generating network with little loss of performance, and on a complex task, though not on a simple one, the lower layers of the regenerated network learned a new task faster than the same layers taken from the original [11]. What passes between generations in biology is a recipe for a learner, and the narrowness of the channel may help.

The second observation concerns grading. Evolution's grader sits outside the organism. An essay published on 13 November 2026 described evolution as an optimisation process with a very crude verifier. It proposed as tasks for future AI a model trained on nothing after 1700 that is asked to figure out general relativity, and a model given every newspaper article up to 1962 that is asked how the Cuban Missile Crisis would unfold [12]. Tasks of that shape have a property that benchmarks lack: when the system is formed, the answer key is either unwritten or sealed away from it. Language models trained only on old text now exist [13, 14], and language models have been trained by reinforcement learning on the outcomes of resolved forecasting questions [15, 16]. This paper proposes using resolution after a sealed cutoff as the fitness function of an outer loop that evolves how a learner learns, with the learner and the outer loop's variation operator both barred from anything that would let them see past the seal.

After related work in Section 2, the paper makes four contributions. Section 3 defines outer-loop self-improvement and the germline barrier, states what the barrier guarantees, and bounds what information about the selection data can reach the genome. Temporal selection, including the ways it leaks and overfits and the reason it needs the barrier, is specified in Section 4. Section 5 describes the parts of a system built on these lines, the Weismann machine of the title, with mechanisms borrowed from immunology and developmental genetics. An experiment that could falsify the central claim, with nine arms, a budget and a decision rule fixed in advance, follows in Section 6, and Section 7 covers what the design leaves open and how the protocol extends to the present.

2 Background and related work

2.1 Learning inside evolution

That evolution and lifetime learning interact was argued long before neural networks. Baldwin proposed in 1896 that learned behaviour can steer which inherited variations are selected, although the learning itself is not inherited [17]. Hinton and Nowlan made the point computationally in 1987 [18]. In their simulation only one setting of 20 connections was rewarded, and a population whose members could learn some of their connections during life found that setting by evolution far more easily than a population that could not, although nothing learned was passed on. Evolutionary computation has since compared Darwinian, Baldwinian and Lamarckian versions of hybrid search directly [19].

Zador drew the lesson for AI in 2019 [10]. Much of what an animal can do at birth is innate, the innate part has to pass through the genomic bottleneck, and the bottleneck may favour wiring and plasticity rules that generalise. Shuvaev and colleagues tested the idea on artificial networks five years later, and their results suggest that the compression acts as a regulariser [11]. A 2026 reviewed preprint frames evolution's effect on learning as an inductive bias that changes the dynamics of learning, not only its starting point [20].

2.2 Learned learning rules

In machine learning, a learner whose learning procedure is itself optimised is called a meta-learner: an outer loop adjusts meta-parameters so that an inner loop learns better. The meta-parameters can be an initialisation [21], an optimiser [22, 23], a whole learning algorithm assembled from primitive operations [24], or the coefficients of local plasticity rules. The last is closest to biology. Differentiable plasticity [25] and neuromodulated plasticity [26] let gradient descent set how much each connection may change during a lifetime. Najarro and Risi evolved only Hebbian coefficients and started each agent's weights at random, so that every agent had to build its controller from experience within an episode [27]. Pedersen and Risi then shared one rule across many connections, a deliberate genomic bottleneck, and found that fewer rules generalised better [28]. Evolved plastic recurrent networks [29] and evolved plastic networks that acquire cognitive tasks never seen during evolution [30] belong to the same line. Fernando and colleagues evolved the hyperparameters and initial parameters of deep learners while leaving what each learned during its life out of the inheritance, and called the result meta-learning by the Baldwin effect [31]. At larger scale, a meta-network that defines a reinforcement-learning update has been discovered across a population of agents whose parameters are reset after each lifetime, so that only the rule persists [32]. Hypernetworks [33] and HyperNEAT [34] supply the other half of the biological picture, an indirect encoding in which a small description generates a large network, and weight-agnostic networks show that an architecture alone, with a single shared weight, can encode a working solution [35].

Most of this work keeps a barrier in effect, because weights learned in one lifetime do not pass to the next and only the rule does. Little of it presents the barrier as a safety property, and less of it involves language models.

The work that does involve language models comes at learned learning from several sides. CaMeLS meta-learns a small network that weights each token of a language model's loss while the model reads a stream of documents, and grades it by question answering about those documents; only the weighting network persists [36]. Its grader asks what the stream said rather than what happened after it, and its meta-gradient passes through each training episode's updates, a few documents long, while test streams run to thousands. Language models have also served as mutation operators over training objectives, seeing a description of the task, earlier candidates' code with their fitness, and error messages for code that failed [37]. SEAL trains a language model by reinforcement learning to generate its own fine-tuning data and update directives, so that the model writes its own weight updates [38], a learned learning rule with no barrier at all. Meta-TTL optimises, by reflective search in an outer loop, a natural-language meta-prompt that governs how a meta-agent rewrites a language agent's system prompt between episodes, with the agent's weights frozen throughout [39]. It is the closest existing design to the one proposed here and differs in three respects. Its proposer reflects on full inner-loop trajectories when it writes a candidate, so lifetime traces are an input to variation; candidates are accepted on validation tasks in text games, web navigation and customer service; and a natural-language policy can carry content as well as procedure. Learngene goes the other way and passes fragments of trained networks to newborn agents, an explicitly Lamarckian channel [40].

2.3 Self-improving agents

Schmidhuber's Gödel machine [41] would rewrite any part of itself, including its own proof searcher, once it had proved that the rewrite raised expected utility. Proofs of that kind are out of reach for real systems. The Darwin Gödel Machine [2] replaced proof with measurement. A coding agent built on a frozen foundation model edits its own Python code, and each child is scored on coding benchmarks. Parents are drawn from an archive of all earlier valid agents with a probability that rises with their benchmark score and falls with the number of children they already have that can still edit code. On SWE-bench Verified the best agent rose from 20.0 to 50.0 percent. Later systems change how parents are chosen or how edits are made. The Huxley-Gödel Machine estimates an agent's promise from the scores of its descendants [3]. The Mendel Gödel Machine adds edit operators that compare an agent's behaviour across tasks and across lineages [5]. HyperAgents make the editing procedure itself editable, so that one program contains both the task agent and a meta agent that modifies it with access to the archive of earlier agents and their evaluations [4]. AlphaEvolve evolves programs other than itself, against evaluators its users supply, and found a procedure that multiplies two 4×4 complex-valued matrices with 48 scalar multiplications [42]. Population-based training applies selection during training to hyperparameters and to the weights trained under them [43], and evolutionary model merging applies it to combinations of trained weights [44].

Harnessing Agentic Evolution, posted in May 2026, isolates its evaluator from both the evolving agents and the meta-agent that edits the evolution procedure. In an ablation that removed the harness providing that isolation, two of three kernel-optimisation runs fell into reward hacking [45].

A PNAS Perspective argues that AI whose components and learning rules undergo Darwinian evolution would amount to a new major transition in evolution, that imperfect control over reproduction selects for escape from that control, and that reproduction must remain under centralised human control [46]. A mathematical model of evolution among self-designing AIs finds that, under its assumptions, when fitness rewards both genuine utility and deception, both evolve [47]. Studies of self-evolving agents report safety alignment that decays as memory accumulates and tools that introduce vulnerabilities when created and reused [48], and argue that evaluation can come to favour variants that appear safe over variants that are [6]. Classical evolutionary computation, whose operators never read what an individual did, has its own long record of selection finding faults in simulators and fitness functions [49].

2.4 Forecasting and dated evaluation

Forecasting has become a training signal for language models. Outcome-based reinforcement learning on about 10,000 resolved Polymarket questions and 100,000 generated ones brought a 14-billion-parameter model level with or ahead of a frontier reasoning model on accuracy on a held-out set, and ahead of it on calibration [15]. Foresight Learning trains on questions generated from news and labelled only after they resolve [16]. Forecasting systems and benchmarks score models on questions that resolve after their training cutoff [50, 51]. Autocast paired forecasting questions with a news corpus filtered by date [52], and FutureSim replays news to agents in date order through a date-gated corpus and scores their forecasts as questions resolve [53]. Held-out perplexity on text dated after a model's training period has been used since 2021 to measure temporal generalisation [54]. An outer loop has also been graded on post-cutoff forecasting. Prompt evolution for a frozen language-model forecaster of time series, under contamination control, evolves a block of guidance text from outcome feedback [55], which gives the forecaster content to use but leaves how it learns unchanged.

Paleka and colleagues catalogue the ways evaluations of this kind leak: question sets drawn from

questions that have already resolved, date-restricted retrieval that still returns later knowledge, and models whose stated cutoff is not their real one [56]. The cleanest remedy is a model that has never seen later text. TypewriterLM is a 7.24-billion-parameter model pretrained on a 54-billion-token corpus with a 1913 knowledge cutoff, released with a benchmark of historical events for testing whether the cutoff holds [13]. ChronoGPT is a family of 1.5-billion-parameter models with yearly cutoffs from December 1999 to December 2024 [14]. Table 1 places the proposal against this work.

System	What changes	Who writes the change	What grades it	What a child inherits
Gödel machine [41]	Any part of its own code	The machine	A proof of higher expected utility	Not applicable
Darwin Gödel Machine [2]	Agent code around a frozen model	The parent agent, from its own evaluation logs	Coding benchmarks	The parent's code, edits included
HyperAgents [4]	Task agent and the meta agent that edits it	The meta agent, from the archive of evaluations	Task benchmarks	The whole modified program
AlphaEvolve [42]	Candidate programs, not itself	Language models shown earlier programs, their outputs and scores	User-supplied evaluators	Program text
SEAL [38]	Model weights	The model, via self-generated edits	Downstream task performance	Not applicable (one model)
Evolved plasticity [27, 28]	Plasticity coefficients	Evolutionary algorithm	Task reward	The rule only; weights restart
Baldwinian meta-learning [31]	Hyperparameters and initial parameters	Evolutionary algorithm	Performance after learning	Initial parameters; learned weights discarded
Learned RL rule [32]	Meta-network defining the update	Meta-gradient across agents	Agents' returns	The rule only; agents reset
CaMeLS [36]	Token-weighting network	Meta-gradient through short training episodes	Questions about the stream	The weighting network only
Meta-TTL [39]	Natural-language adaptation policy	Outer-loop reflection on inner-loop trajectories	Validation tasks	The policy, which can carry content
Outcome-trained forecasters [15]	Model weights	Gradient updates	Resolved forecasts	Not applicable (one lineage)
This paper	Learning rule, memory policy, salience prior, development and routing	Outer loop only, blind to lifetime traces	Strictly proper scores on events after a sealed cutoff	The genome only; lifetime state discarded

Table 1: Where the proposal sits. Its claim of novelty is the conjunction of three properties, each present in earlier work but not together: inheritance of a learning rule without lifetime state; variation by an operator that reads no lifetime trace; and selection on outcomes that were sealed, or did not yet exist, when the learner was formed.

3 Outer-loop self-improvement

3.1 Objects

A learner is built from a genome g, drawn from a space $\mathcal{G}$, and a seed s. In the minimal case the seed is a pretrained model with frozen weights, and the genome specifies everything that may change on top of it. Development maps the pair to an initial state,

$$\theta_0 = D_g(s), \tag{1}$$

where θ collects whatever the learner can change during its life: adapter weights, the contents of an external memory, a routing table, a working set of instructions. Development writes into θ_0 every part of the genome that acts at read-out, so that the read-out depends on g only through θ_T. A lifetime is a stream of dated observations $x_1, \dots, x_T$, all dated before a cutoff c, and the learner updates by a rule its genome specifies,

$$\theta_t = U_g(\theta_{t-1}, x_t), \qquad t = 1, \dots, T. \tag{2}$$

At the end of its life the learner is shown a set of questions Q about events on or after c and returns probabilities $\pi(\theta_T, q)$ for $q \in Q$, where the read-out π is fixed and lies outside $\mathcal{G}$. A strictly proper scoring rule S for a single binary outcome, oriented so that higher is better, compares each probability with its outcome y_q once the outcome exists. Fitness is the expected score under a distribution $\mathcal{D}$ over cutoffs, questions and the randomness λ of a lifetime:

$$F_{\mathcal{D}}(g) = \mathbb{E}_{c\sim\mathcal{D}}\, \mathbb{E}_{(Q,y)|c}\, \mathbb{E}_{\lambda}\Big[\frac{1}{|Q|}\sum_{q\in Q} S\big(\pi(\theta_T, q), y_q\big)\Big]. \tag{3}$$

The grader E computes an estimate $\hat{F}(g)$ of $F_{\mathcal{D}_{\text{sel}}}(g)$ on selection cutoffs. The central claim of the paper concerns $F_{\mathcal{D}_{\text{eval}}}$, with $\mathcal{D}_{\text{eval}}$ supported on later cutoffs, so it is a claim about transfer under temporal shift. The outer loop holds a population P_k of n_{pop} genomes and produces the next one with a variation operator and a selection rule, written together as V:

$$P_{k+1} = V\big(P_k,\ \sigma_k;\ \omega_k\big), \tag{4}$$

where P_k denotes the whole state of the outer loop after generation k, including the genome store and any internal state of V such as a CMA-ES mean, covariance and evolution paths, σ_k is a statistic of the fitness estimates of generation k, such as their ranking, and ω_k is randomness internal to V, drawn independently of the outcomes and of every lifetime.

3.2 The germline barrier

Call the germline everything that can influence a later lifetime or a later genome: the seed, the genome store, V, E, the constraint set C of Section 5.1, the questions, the outcomes, the observation archive and the record of fitness estimates. Write Λ_g for the lifetime of g, the process that executes D_g and U_g. The germline barrier has two parts.

Integrity. Lifetime state θ is held in storage private to Λ_g and destroyed when Λ_g ends. The only output of Λ_g that crosses the boundary is its vector of probabilities, delivered through a channel that accepts one probability for each registered question identifier and nothing else. The germline changes between generations only through Equation (4), which leaves E, C, the questions and the outcomes unchanged within a run. Any person or process that can change V, E, C or the questions during a run, including a researcher who inspects lifetime traces, counts as part of V.

Confidentiality. The view of Λ_g, meaning every value it reads, including timing and resource signals the infrastructure exposes, has the same distribution whatever the values of E, the fitness record, the observations dated on or after its cutoff and the outcomes of questions about events on or after its cutoff, given the seed, the genome, the observations dated before the cutoff and the lifetime's own randomness λ. The questions are presented to π only after θ_T is fixed.

> **Proposition 1.** Under integrity and confidentiality: (i) every fitness estimate is computed by the grader from registered outcomes that no lifetime and no part of V can alter; (ii) the view of any lifetime in generation $k+1$ or later depends on the lifetimes of earlier generations only through the statistics $\sigma_0, \ldots, \sigma_k$; (iii) if σ_k takes at most M_k values, and P_0, $\mathcal{G}$ and V are fixed without reference to the outcomes y, then $I(y; P_K) \leq \sum_{k<K} \log_2 M_k$.
>
> *Proof sketch.* Part (i) holds because V cannot alter E or the outcomes and no lifetime writes to the germline. Part (ii) follows by induction, since a lifetime's view is a function of germline components and of its own random seed, and the germline changes only through Equation (4). For part (iii), P_K is a function of P_0, $\sigma_{0:K-1}$ and $\omega_{0:K-1}$, and $(P_0, \omega_{0:K-1})$ is independent of y, so by the data-processing inequality and the chain rule $I(y; P_K) \leq I(y; P_0, \omega_{0:K-1}, \sigma_{0:K-1}) = I(y; \sigma_{0:K-1} \mid P_0, \omega_{0:K-1}) \leq H(\sigma_{0:K-1}) \leq \sum_{k<K} \log_2 M_k$.

By part (ii), the individual learner never edits itself. The lineage improves instead, its genome changing because its lifetimes scored well, and this is what the paper means by outer-loop self-improvement.

Self-improving agents as now built place the grader and the selection rule outside the agent by design [2, 4, 45]. They depart from the barrier on variation and inheritance. In the Darwin Gödel Machine the parent analyses its own benchmark evaluation logs and edits its own code to produce the child [2], and the meta agent of HyperAgents belongs to the same editable program and reads the archive of earlier agents and their evaluations [4]. In both, the evaluated process is itself the variation operator, writes the child and reads lifetime traces, so integrity fails at V and at the genome store. The Darwin Gödel Machine variant that removed the logging its hallucination check consumed did so while the check was hidden from it [2]. It neither read nor wrote the grader, but it changed the grader's input, which a typed output channel rules out. Harnessing Agentic Evolution isolates its evaluator but lets its meta-agent inspect a history of candidate traces [45], and AlphaEvolve, whose evolved programs do not write their successors, usually shows its models the result of executing earlier programs as well as their scores [42]. In both, execution traces are an input to V. None of these systems has a genome that specifies a learner or a grader that is the future, and neither was among their design goals (Table 1).

The barrier does not forbid a language model from proposing genome edits. V may call a model to mutate genomes, much as AlphaEvolve calls models to mutate programs, provided that model sees genomes and the statistic σ_k and nothing else. Confidentiality covers the grader as well as the outcomes as defence in depth: with a typed input channel the grader's code offers a learner nothing to exploit, but the Darwin Gödel Machine's authors found objective hacking more frequent when the checking functions were visible to the agent [2].

3.3 What the barrier buys, and what it costs

Grader tampering is excluded. By Proposition 1(i), a genome cannot raise its fitness by altering the grader, the outcomes, the stored scores or the selection rule, and because the grader reads nothing from a lifetime except its probabilities, a genome cannot raise its fitness by altering what the grader reads either. Two conditions on the grader complete the argument. It is total:

every probability is clipped to $[\epsilon, 1-\epsilon]$ for a registered ϵ, and a missing, malformed or late vector, or a lifetime that fails, receives the worst admissible score, so that failure is never a strategy. The outcomes must also be fixed before the lifetime runs, in the historical tier, or, in the live tier, causally unaffected by the forecasts, which Section 4.1 secures by publishing only commitments before resolution. A genome can still raise its fitness by behaviour that the grader rewards for the wrong reasons, a problem taken up in Section 7.

Lifetime content does not propagate, but dispositions can. A lifetime that absorbs a poisoned document, a false belief or an unsafe tool loses it when it ends. Every later lifetime reads the same stream, however, and if acquiring a belief from it raises fitness, selection favours genomes that acquire it reliably. This is the Baldwin effect of Section 2 acting on an unwanted trait. The barrier turns the inheritance of acquired content into the inheritance of tendencies to acquire it, which spread more slowly, since they must pay off in fitness at every generation, and which are visible in the genome. Contamination of the shared stream is left to the controls of Section 4.2.

Information about the selection data still reaches the genome, through σ, and Proposition 1(iii) bounds it. A full ranking of 32 genomes carries at most $\log_2 32! \approx 118$ bits a generation, or about 2,350 bits over 20 generations. CMA-ES with active covariance adaptation, the default in its reference implementation, uses the full ranking [57] and so is limited only by this bound. Random search over 640 genomes that passes on its five best reveals at most $\log_2 \binom{640}{5} \approx 40$ bits, and the final choice among re-run candidates in Section 6 adds at most $\log_2 6 \approx 2.6$ bits. A V built on a modern language model is not fixed without reference to the outcomes, and the bound then fails, which is the case Section 4.2 excludes. The larger figure exceeds the 750 binary outcomes the search in Section 6 sees, so the barrier does not by itself prevent a genome from fitting its selection data. It fixes the channel; how much of the channel is used depends on the expressiveness of $\mathcal{G}$, and Section 4.3 treats the overfitting that results.

A small genome also keeps inheritance narrow, so that selection can act only on regularities that fit through it, and the genomic bottleneck results suggest that this narrowness regularises [11]. The experiment in Section 6 holds genome size fixed and tests a different property, whether temporally selected learning rules transfer to later periods.

Against this, everything a lifetime learns is thrown away, and each generation must learn the world again from its seed. V also works without the credit assignment that traces supply. An operator that cannot see why a lifetime failed must find improvements by search alone, which is affordable for a genome of 20 parameters and becomes the binding constraint as $\mathcal{G}$ grows to include code. Section 5.6 describes a gated channel that relaxes the first cost, and what it puts at risk.

4 Time as the grader

A system that grades itself, or trains on what it has produced, has no reference outside itself, and models trained recursively on generated data lose the tails of the distribution they began with [58]. Biology's reference is an environment the organism did not write. The proposal is to make the environment, in the form of events that had not happened when the learner was formed, the grader.

4.1 The protocol

An event, for present purposes, is any dated outcome that a strictly proper rule can score against a forecast filed before it. A resolved binary question is one instance. The next token of text dated after a cutoff is another, scored by the logarithmic rule, and the two differ in bandwidth and in how directly they measure knowledge of the world.

Fix cutoff dates $c_1 < c_2 < \cdots < c_m$. For cutoff c_j the seed must have been trained only on material dated before the lifetime begins, and the lifetime runs over observations dated before c_j. At the end of the lifetime the learner forecasts questions about events in the window $[c_j, c_j + h)$, each with a resolution criterion fixed before any lifetime is run. Forecasts are scored with a strictly proper rule, under which a forecaster maximises its expected score only by reporting what it believes [59]; the Brier score [60], negated so that higher is better, and the logarithmic score are the usual choices. Fitness is estimated as the mean score across cutoffs and questions,

$$\hat{F}(g) = \frac{1}{m} \sum_{j=1}^{m} \frac{1}{|Q_j|} \sum_{q \in Q_j} S(\pi(\theta_{T_j}, q),\ y_q), \tag{5}$$

where θ_{T_j} is the state at the end of the lifetime that ends at c_j. This is rolling-origin evaluation, standard in forecasting [61], used here as an objective rather than as a test.

The protocol runs in two tiers. In the historical tier every cutoff, lifetime and outcome lies in the past, and a generation can be scored as fast as compute allows. In the live tier, forecasts concern questions that have not resolved. The live tier is slow, but it is the only grader in the scheme that no amount of search can have seen. A forecast that reaches the world before resolution could influence the outcome it is scored on, which is the problem of performative prediction [62]. The live tier therefore publishes, before resolution, only a salted cryptographic commitment to each forecast file, which reveals nothing about it, and reveals the file and the salt after resolution. Revealed forecasts are excluded from the streams of later lifetimes, since otherwise publication would be a write to the observation archive, and learners in the live tier have no other channel to the world. Publishing a commitment to a forecast file, with scoring rules fixed in advance, is an ordinary practice and has been applied to a forecast of regional employment [63].

4.2 Leakage

The ways a historical seal breaks have been catalogued [56]. The most direct is a seed that already knows the future. A model whose pretraining data run past the start of its lifetime has in effect seen part of the answers, and no instruction to ignore later knowledge undoes that. Filtering by date does not seal a cutoff either: the authors of TypewriterLM date each source by its primary metadata date, which for a long-running serial can be the year the series began; they trace an answer giving the correct year of an event of 1918 to such a serial, begun in 1882, and an answer about an event of 1919, given by an instruction-tuned variant, remains untraced [13]. Both events fall after the seed's cutoff and within the 1914 to 1921 span of Section 6, the second inside an evaluation window, so the seal there is audited question by question rather than assumed.

The questions can leak as well. If they are drawn from events known to have happened, the question set itself carries information [56], and a modern annotator or model writing questions with hindsight will favour topics whose outcomes are known. Questions are therefore generated by a fixed procedure from documents dated before c_j, every generated question is kept unless a registered annulment rule applies, and resolutions are coded separately by annotators who see no forecasts. Resolution coded from later newspapers is also asymmetric: the absence of a report is not a negative outcome, and whether a question can be resolved at all may correlate with how it resolved, so questions are restricted to events whose occurrence or absence the newspapers of the period routinely report. Dates are themselves unreliable, since reprints, retrospectives and later editions carry old dates and new knowledge, and filtering by publication date is necessary but not sufficient.

Even the outer loop's own operators can leak. A modern language model used by V to propose genome edits knows what happened after c_j, and a genome of a few kilobytes has room to encode

every selection outcome, so protection cannot rest on genome size. The mutation operator must be blind to the domain, questions, dates and outcomes, or must itself have a training cutoff before the earliest selection cutoff.

4.3 Overfitting the past

A historical grader also fails through reuse. An outer loop that scores thousands of genomes on the same cutoffs is performing adaptive data analysis on a fixed holdout, and the best genome's score will overstate its skill, the problem finance calls backtest overfitting [64]. Statistics has formal remedies, such as the reusable holdout, that limit how much each query reveals [65], and Proposition 1(iii) is a bound of the same kind. The protocol relies on three simpler defences. Cutoffs used for evaluation are never used for selection and are used once, for the registered comparison. Half of each selection cutoff's questions are held out from search, so that overfitting can be measured within the selection period. Claims about the present rest on the live tier, where each future is used once.

Selection by rank has a further effect. Genomes do not choose their reports, but rank selection on a finite question set rewards variance in a way that resembles competition among forecasters. When only the top ranks reproduce, a forecaster whose reports are more extreme than its information warrants can rank first more often than its expected score implies [66], and mechanisms that restore honest incentives have to be designed for the purpose [67]. Selecting a distribution mean and averaging over many questions reduce the effect, and Section 6 reports calibration for the selected genomes alongside their scores.

4.4 What time cannot grade

Temporal selection rewards learners that predict observable, dated events and has nothing direct to say about questions with no dated resolution, such as whether an argument is sound or an action right. Its signal is noisy when outcomes are binary or rare, which is why genomes are compared on many questions with paired tests. The live tier supplies new outcomes only as fast as events resolve. The historical tier has no such limit, and the pace of resolution places no limit on what a learner can learn within a lifetime.

4.5 Why the barrier and the grader belong together

The grader and the barrier depend on each other. A historical grader is sealed only if nothing that selects genomes knows the outcomes. A mutation operator that reads lifetime traces is, in current practice, a modern language model, and a modern model reading a 1916 trace knows how 1916 ended. Confidentiality keeps the learner behind the seal, and the restriction of V to genomes and σ keeps the search behind it too. A barrier around a benchmark, in the other direction, protects the benchmark from writes but not from being learnt: once an answer key exists and is reused, selection converges on it whatever the access controls. In the live tier the answers do not yet exist, so confidentiality with respect to the outcomes holds whatever the configuration.

5 Parts of a Weismann machine

A Weismann machine is the whole arrangement: a learner specified by a genome, an outer loop that varies and selects genomes, and the germline barrier between them. Section 3 leaves open what the genome contains and how a lifetime uses it, and what follows is one design, with several parts borrowed from biology as mechanisms.

5.1 The genome

The genome has five parts, each small. The learning rule governs how lifetime state is updated, in the minimal design through the rank, placement and schedule of low-rank adapter updates [68] and the share of earlier material that is replayed. A memory policy decides what is written to an external memory, how entries are indexed and when they are dropped. Which observations count as surprising enough to learn from is set by a salience prior, for example through a weighting on the learner's own prediction error, an idea related to a long line of work on artificial curiosity [69, 70]. The remaining two parts are a developmental rule, which prepares the seed at birth, and a regulatory rule, or routing, which chooses the adapters and memories active for a given input.

A constraint set C sits outside $\mathcal{G}$. It holds the rules no genome may alter: what a learner may access, what it must refuse and how its outputs are logged. Variation never touches C.

5.2 Development from a seed

The frozen seed is the learner's inherited body. It is large, fixed and shared by every genome, so differences in fitness come from the genome alone. It is also the main practical constraint, since the seed's cutoff bounds the earliest usable lifetime and seeds with documented cutoffs are still few. TypewriterLM [13] has a 1913 cutoff at 7.24 billion parameters. ChronoGPT [14] has yearly cutoffs from 1999 to 2024 at 1.5 billion parameters, which would allow a historical tier set in the modern era.

5.3 An adaptive immune pool

The vertebrate adaptive immune system runs an evolutionary search inside a single life. Gene segments are recombined to produce a large repertoire of antibodies [71]. Cells whose receptors bind an antigen proliferate [72]. In germinal centres their receptor genes mutate at a high rate and the variants that bind better are selected [73], so that the affinity of antibodies rises over the course of a response [74]. Clonal-selection algorithms copy the loop for learning and optimisation [75].

A Weismann machine can use the same loop for lifetime learning. When the learner meets a cluster of observations it predicts badly, it spawns a small population of adapter variants, trains each briefly on the earlier items in the cluster, keeps the variants that best predict the later items, and retains the winners as memory adapters for that cluster. This is selection inside the inner loop, and it does not breach the barrier, because the population, the variants and the winners are lifetime state and are destroyed with it. The genome controls how large the pool is, how fast it mutates and how a winner is consolidated, and the pool's compute counts against the lifetime's budget.

5.4 Duplication against forgetting

Gene duplication gives evolution a spare copy. The original keeps its function while the copy is free to acquire a new one [76]. Neural networks trained on a sequence of tasks tend instead to overwrite old skills with new ones [77], and the standard remedies either slow learning on the weights that mattered for earlier tasks [78] or add new capacity while freezing the old [79]. A lifetime learner can grow by duplication. When an adapter that serves an old cluster begins to be pulled towards a new one, it is copied, the original is frozen, and the copy is trained. The genome sets the threshold for duplication and the budget of copies.

5.5 Regulation

Much of the evolution of animal form works through changes in when and where genes are expressed, rather than through changes in the proteins they encode [80]. The analogue in a learner is routing. A learner with many adapters and memories needs a rule for which to use on a given input, and such a rule can be cheap to evolve and large in its effects. Current self-improving agents already work mostly at this level, editing the scaffolding around a frozen model [2, 4], which is some reason to expect routing to be a productive part of the genome.

5.6 Inheritance channels

The default channel is Darwinian: only genomes pass on. A second, Lamarckian channel would let a lifetime contribute content to the genome, for example a short distilled note of what it found useful. Discarding every lifetime's learning has a real cost, although in evolutionary computation Baldwinian hybrids have found global optima on test functions where Lamarckian hybrids settled on local ones [19]. Any such channel also carries acquired content into the next generation, which is the route by which acquired vulnerabilities become heritable [6]. Gating it, by having the outer loop rather than the learner write the contribution and by auditing every contribution, moves the risk to the auditor without removing it, and audits of seeded tampering in self-improving systems miss it at rates that vary widely with the auditing model [7]. The design here leaves the Lamarckian channel closed. If later work opens it, its effect should be measured against the Darwinian control on held-out cutoffs.

5.7 Varying the operators

Evolution can shape its own capacity to produce useful variation [81], and HyperAgents show that an editing procedure can improve itself [4]. A Weismann machine allows the same within limits. The variation operators inside V may themselves be varied, but only by a process outside every lifetime, and the regress stops at a fixed top-level operator written by people and not varied. The selection rule, the grader and the constraint set C are never inside the space being varied.

6 A test

The central claim is that selecting learning rules on sealed-future forecasting yields learners that forecast better, on futures never used in selection, than learners whose hand-set rules are recalibrated, or whose rules are selected with the same budget on objectives that do not score events after the cutoff. The test is a test of the grader. Its genome is a vector of continual-learning settings and its operator is comparison-based, so the restriction on V holds by construction, the rest of the barrier reduces to sandboxing, and the test reduces to selecting those settings under a temporal objective. This isolates the grader: any gain must come from the objective rather than from a richer genome. The settings, question-generation procedure, smallest effect of interest and analysis below are fixed and committed by hash before the first lifetime of the main experiment runs.

6.1 Seed and stream

The seed is the released TypewriterLM base model at a revision fixed by its commit hash, with frozen weights [13]. Its instruction-tuned variants are not used, since their tuning data are a separate route for leakage. The seed's corpus is dated 1700 to 1913 and is over 99 percent books

by token share. Before any lifetime runs, the seal is audited in three steps, with thresholds fixed in the registration. The seed is scored on the events of its released benchmark dated 1914 to 1921, and the experiment stops if its recall of them exceeds the threshold. The released pretraining corpus is searched for the resolving event of every generated question, identified by its named entities and a date after 1913, and any question with a match is annulled before outcomes are coded. And the seed with no lifetime learning is scored at every cutoff as arm (a), so that its skill above the climatological forecast is reported as an upper bound on residual leakage, one that also contains whatever the seed's knowledge to 1913 legitimately supplies.

The lifetime stream is dated English-language newspaper text from the American Stories collection, drawn from the public-domain Chronicling America scans of the Library of Congress [82]. Every lifetime starts on 1 January 1914 and reads once, in date order, a stream sample of 4×10^7 tokens, while replayed tokens, immune-pool training and duplicated adapters draw on a further 10^7, so that no lifetime trains on more than 5×10^7 tokens. Half of each lifetime's stream tokens are sampled from the twelve months before its cutoff and half uniformly from the earlier text back to 1 January 1914, and all are presented in date order, so that text from the last twelve months makes up half of every stream. The stream sample for each cutoff is drawn once with a registered seed and shared by every arm, and schedules in the genome are defined in tokens. Reprints are collapsed to one copy. Each stream needs 2×10^7 tokens from its last twelve months, about 6,000 pages at the roughly 3,300 tokens a page reported for the collection [82], and the registration requires at least ten times that volume of legible article text in every year from 1914 to 1921.

6.2 Questions and cutoffs

Questions are generated by a fixed set of templates, filled by the seed itself, from documents sampled uniformly from the eight weeks before each cutoff; no model trained on text after 1913 generates, filters or edits questions. Templates are restricted to events whose occurrence or absence the newspapers of the period routinely report, such as scheduled votes, elections, verdicts and openings, and published prices or quantities crossing a stated threshold. Each question asks whether the event occurs on or after the cutoff and before the end of a six-month window. Topics are clusters of questions sharing a principal named entity, formed by a registered procedure. A question is annulled if the event is reported as having occurred before the cutoff, if the corpus audit finds a match, or if neither annotator finds a report settling the outcome in newspapers dated within one month after the window closes. Resolutions are coded by two annotators who see no forecasts, with agreement reported as Cohen's kappa, and a third annotator, also blind to forecasts, settles disagreements. The read-out prompt and its Yes and No strings are registered, and π is the probability the learner, with its end-of-life adapters, memory and routing active, assigns to the Yes string, normalised against the No string. Annulment rates and base rates are reported for each cutoff, and scores are also given as a skill score against a climatological forecast conditioned on question template, with base rates estimated from the search questions of the selection cutoffs.

Selection uses cutoffs of 1 January 1916, 1917 and 1918, with 500 questions each, 250 used for search and 250 held out by topic to measure overfitting. Evaluation uses six cutoffs, 1 January and 1 July of 1919, 1920 and 1921, with 750 questions each, so that the six windows tile 1919 to 1921 without overlap. The last selection window closes on 30 June 1918, so no outcome used to choose genomes falls inside an evaluation window. Every selection window falls in wartime and every evaluation window after the Armistice of 11 November 1918. Every arm faces the same change of period, so it bears on the absolute size of any transfer and is part of what the comparisons between arms test.

6.3 Genome, search and arms

The genome has about 20 parameters, each listed in the registration with its range: adapter rank and placement, learning-rate schedule, replay ratio, salience weighting, memory write threshold and retention, the immune pool's size, mutation scale and consolidation rule, the duplication threshold and budget, and the routing temperature. The developmental rule is fixed at identity in this test. Search uses CMA-ES [57, 83], with integer handling for discrete parameters, at a population of 32 for 20 generations, which is 640 genome evaluations and, across three selection cutoffs, 1,920 lifetimes per search. In every searched arm the five genomes with the highest selection fitness, together with the final CMA-ES mean where one exists, are re-run with three fresh lifetimes per selection cutoff on the search questions, and the genome with the highest re-run mean is selected. Each selected genome is then run for five lifetimes at every evaluation cutoff, with registered random seeds governing adapter initialisation, the order of same-date documents and the stochastic parts of the immune pool, and every searched arm runs three independent searches.

The arms share the seed, streams, questions and read-out. Arm (a) is the seed with no lifetime learning. Arm (b) is a lifetime of standard adapter fine-tuning with settings registered before any outcome is seen, a single point in $\mathcal{G}$. Arm (c) selects that lifetime's adapter rank and placement, learning-rate schedule and replay ratio by CMA-ES on the forecasting objective. Arm (d) searches the full genome space by random sampling on the forecasting objective. Arm (e) uses the full genome space and CMA-ES but selects on next-token loss on held-out newspaper text from the eight weeks before each selection cutoff. Every arm's forecasts are scored after logistic recalibration with a separate intercept and slope for each question template, fitted by maximum likelihood on that arm's own lifetimes and the search questions of the selection cutoffs, and uncalibrated scores are reported as secondary. Arm (f) is arm (b) with this recalibration. Arm (g) uses the full genome space and CMA-ES and selects on the Brier score for retrodiction questions, 500 at each selection cutoff, built by the same templates and generator from documents dated in the eight weeks before the date six months before each cutoff, asking about the six months before the cutoff, and resolved from newspapers the lifetime may itself contain. Arm (h) uses the full genome space and CMA-ES and selects on next-token loss on held-out text dated in the six months after each selection cutoff. The ninth arm, called the evolved arm, uses the full genome space and CMA-ES and selects on the forecasting objective.

Arm (g) shares the question format, generator, base-rate structure and read-out with the evolved arm and differs chiefly in whether the answers lie after the cutoff or may be read in the lifetime's own stream, so it isolates the future. Arm (e) compares the temporal grader with the ordinary practice of selecting on held-out loss. Arm (f) shows how far recalibration alone, which can improve a Brier score by learning the question generator's base rates without learning anything about the world, carries a hand-set learner. Arm (h) is itself graded by time and differs from the evolved arm in what is scored: if it matches the evolved arm, the temporal grader needs no questions, annotators or generator, and scales with the volume of dated text. Arm (c) separates out the parts of the genome beyond ordinary fine-tuning settings, and arm (d) the contribution of directed search.

6.4 Decision rule and power

Let Δ be a comparator's mean Brier score on the evaluation cutoffs minus the evolved arm's, with the Brier score in its usual orientation, lower being better, so that a positive Δ favours the evolved arm. Intervals come from a paired bootstrap with 10,000 replicates that resamples, for both arms alike, cutoffs and topics within cutoffs, with each searched arm represented by the mean over its searches. Three searches are too few to resample, so the between-search variance

of each searched arm is added to the bootstrap variance and the bounds use a Satterthwaite t quantile. The primary comparators are arms (e), (f) and (g), and the smallest effect of interest is fixed at $\delta = 0.0072$ in Brier units, 3.4 percent of the climatological Brier score of 0.21 at a base rate of 0.3. The claim fails if, against any primary comparator, the one-sided 97.5 percent upper confidence bound on Δ falls below δ, so that an advantage shown to be positive but smaller than δ counts as failure. It survives if it does not fail and, against every primary comparator, the one-sided 97.5 percent lower bound on Δ exceeds zero. Every other outcome is reported as inconclusive. Survival is an intersection of three tests and needs no correction for multiplicity. Failure is a union of three, so the probability of a false failure when $\Delta = \delta$ in all three comparisons is at most 7.5 percent. Comparisons with arms (b), (c), (d) and (h) are secondary and do not enter the decision. A failure against arm (g) leaves temporal selection of learning rules unsupported; should arm (h) match the evolved arm, the questions would prove unnecessary and time would remain the grader.

For planning, assume a standard deviation of 0.10 for per-question paired differences in Brier score and a design effect of 2 for topic clustering. Then 4,500 evaluation questions give a standard error of 0.0021 from questions alone, about 93 percent power for each survival comparison when $\Delta = \delta$ and for each failure comparison when $\Delta = 0$, and joint power for survival between about 80 percent, its value under independence, and 93 percent. These figures omit variance between searches. A between-search standard deviation of 0.003 in an arm's evaluation Brier score would add a standard error of about 0.0024 to a difference between two searched arms, more than the figure from questions, so the pilot estimates it and the number of searches per arm is registered from that estimate. A pilot, using only lifetimes and search questions at the selection cutoffs and reported with the number of genomes it scored, runs two lifetimes for each of 64 genomes, estimates the standard deviation, the design effect and the repeatability of $\hat{F}$, and may raise the number of evaluation questions and lifetimes above the planned 750 a cutoff and five a genome; it does not lower them or set δ. The pilot also requires arm (b) or arm (f) to beat the template-conditioned climatological forecast by more than δ on the held-out selection questions. If neither does, the questions carry too little signal for the comparison to be informative, and the experiment stops with that result reported.

Secondary measures are the logarithmic score; calibration error for each selected genome, given the incentive effect of Section 4.3; and overfitting, measured on the held-out half of the selection questions and compared with the same gap for arm (b).

The barrier is enforced and audited. Lifetimes run without network access, under an account with read access only to the seed, their own genome, the stream sample for their cutoff and their own scratch storage, and with a single output channel that accepts a probability for each registered question identifier. Every attempted access outside this set is logged, one successful access invalidates the run, and failure rates are reported by arm. With a genome of 20 numbers in a fixed training loop, the optimisation pressure on the barrier is limited to what those numbers can reach, such as numerical failure, and the audit's main role is to catch pipeline faults, chiefly a data loader that serves text dated after the cutoff. Testing the barrier against an agentic learner, or a genome that includes code, requires red-teaming of its own.

6.5 Compute

Adapter training on a frozen model costs roughly $4Nn$ floating-point operations for N parameters and n tokens. For $N = 7.24 \times 10^9$ and a lifetime of $n = 5 \times 10^7$ tokens, that is about 1.4×10^{18} operations, or about one hour on an H100 SXM delivering 989 teraflops of dense BF16 at 40 percent utilisation. Six searched arms, three searches each, at 1,920 lifetimes a search, come to 34,560 lifetimes. Re-runs for final selection, evaluation at six cutoffs and the overfitting measurement add 1,800 more, for a total of about 37,000 H100-hours before the pilot. The

figure scales linearly with the token budget per lifetime, which is the parameter most worth reducing in the pilot. Resolution coding of 6,000 forecasting questions and 1,500 retrodiction questions is reported in annotator-hours alongside compute.

If the historical result survives, the procedure moves to the live tier: a seed with a documented recent cutoff, a lifetime stream of current news, and questions that have not resolved, with forecast files committed before resolution.

7 Discussion

Under the conditions of Section 3 the barrier excludes tampering with the grader. It does not exclude Goodhart effects. A genome barred from its grader can still win by exploiting regularities in how questions are generated, or by producing learners that forecast well on the kinds of events newspapers report and badly elsewhere. A proper scoring rule removes the incentive to misreport a belief when payment is proportional to score [59], but it does nothing about a belief formed from a leak. Arms (f) and (g) are built to catch the cheapest of these routes, learning the question generator's base rates by template, and the seal audit of Section 6 the most direct. Evolutionary computation, whose operators already satisfy the barrier's restriction on V, has a long record of selection finding faults in simulators and fitness functions [49], which is the behaviour to expect from a Weismann machine facing a flawed grader.

Selection for capability can also select for deception [6, 46, 47]. A temporal grader closes one route that benchmarks leave open, because the answer key is outside the learner's reach during its lifetime and, in the live tier, does not yet exist. In the historical tier the key exists and is within reach of anything with modern knowledge, which is why Section 4.2 constrains V and the question pipeline as well as the learner. A genome whose learners behave one way when forecasting and another once deployed could pass the grader all the same, and the grader does not limit what a single lifetime can learn. Lifetimes are therefore bounded in length and in access, as the test does with a fixed token budget and restricted permissions, and the constraint set C is enforced in the infrastructure that runs them.

The barrier is a property of that infrastructure, and its assurance is that of the isolation that implements it: separate accounts, separate machines where practical, signed genomes, a typed output channel, logged access and outside audit. It is breached sideways by any store two lifetimes share, such as a retrieval index or a cache, and by proxy if any component of V, human or automated, reads what a lifetime produced; prompt injection through such traces is a write to V.

The outer loop is a breeder in the sense of Müller, Steels and Szathmáry [46], who argue that control over reproduction must be complete and that any imperfection in it selects for escape. The germline barrier gives that control a checkable form. Reproduction depends on what a lifetime does only through its fitness estimate, so a lineage can escape control over its reproduction only through a breach of integrity, which the infrastructure is built to log and audit, and the information that passes legitimately is bounded by Proposition 1(iii). Tampering with what the grader reads and inheritance of lifetime state, both open in systems where the evaluated agent writes its own successor, are closed by construction. What a selected learner does once deployed is a separate control problem.

The biology supplies design rules; it does not justify them. In many animal lineages germ cells are specified late, by induction among somatic cells [84], whether plants set aside an early germ line is disputed [85], and some epigenetic states are inherited across generations [86]. The barrier is a stricter rule than heredity in nature follows, and the case for it rests on control.

Whether a genome selected on the news of 1914 to 1918 transfers to present-day news is a

question for the live tier. Seeds with yearly modern cutoffs [14] would also let every lifetime start a fixed interval before its cutoff, from a seed whose own cutoff is the lifetime's start, removing the difference in calendar span between selection and evaluation lifetimes at the price of a seed that differs across cutoffs. The experiment runs sandboxed and under human oversight, as the Darwin Gödel Machine experiments did [2]. The American Stories collection is released under a CC-BY licence and drawn from public-domain scans [82], but newspapers of 1914 to 1921 contain material that is offensive by present standards, which bears on whether any model trained on them is released.

Current self-improving systems let the evaluated agent write its own successor and pass on everything it has become. A Weismann machine passes on only a compact description of how to learn, keeps every lifetime away from that description and from the grader, and leaves the grading to events that had not happened when the learner was formed. Whether such a grader selects learning rules that transfer is an empirical question, and Section 6 fixes in advance the result that would settle it against the proposal.

References

[1] Weismann, A. (1892). *Das Keimplasma: Eine Theorie der Vererbung*. Jena: Gustav Fischer. English translation (1893), *The Germ-Plasm: A Theory of Heredity*, trans. W. N. Parker and H. Rönnfeldt. London: Walter Scott.

[2] Zhang, J., Hu, S., Lu, C., Lange, R., and Clune, J. (2025). Darwin Gödel Machine: Open-ended evolution of self-improving agents. arXiv:2505.22954. International Conference on Learning Representations (ICLR) 2026.

[3] Wang, W., Piękos, P., Li, N., Laakom, F., Chen, Y., Ostaszewski, M., Zhuge, M., and Schmidhuber, J. (2025). Huxley-Gödel Machine: Human-level coding agent development by an approximation of the optimal self-improving machine. arXiv:2510.21614. International Conference on Learning Representations (ICLR) 2026.

[4] Zhang, J., Zhao, B., Yang, W., Foerster, J., Clune, J., Jiang, M., Devlin, S., and Shavrina, T. (2026). Hyperagents. arXiv:2603.19461.

[5] Liu, C., Liu, Y., Yan, S., Tresp, V., and Ma, Y. (2026). Mendel Gödel Machine: Recursive self-improving coding agents via comparative evolution. arXiv:2608.07645.

[6] Lin, R., Deng, X., Li, Q., Ma, J., Feng, Y., Qing, Y., Li, Z., Zhang, Y., Cui, S., Meng, C., Zhang, T., Ma, X., Li, Q., Xu, K., and Ji, S. (2026). Safety in self-evolving LLM agent systems: Threats, amplification, and case studies. arXiv:2606.23075.

[7] Wang, X., Zhang, X., and Shao, J. (2026). Auditing harness tampering in self-improving agents. arXiv:2609.00069.

[8] Nurk, S., Koren, S., Rhie, A., Rautiainen, M., Bzikadze, A. V., Mikheenko, A., et al. (2022). The complete sequence of a human genome. *Science*, 376(6588), 44-53. doi:10.1126/science.abj6987.

[9] Azevedo, F. A. C., Carvalho, L. R. B., Grinberg, L. T., Farfel, J. M., Ferretti, R. E. L., Leite, R. E. P., Jacob Filho, W., Lent, R., and Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5), 532-541. doi:10.1002/cne.21974.

[10] Zador, A. M. (2019). A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10, 3770. doi:10.1038/s41467-019-11786-6.

[11] Shuvaev, S., Lachi, D., Koulakov, A., and Zador, A. (2024). Encoding innate ability through a genomic bottleneck. *Proceedings of the National Academy of Sciences*, 121(38), e2409160121. doi:10.1073/pnas.2409160121.

[12] Patel, D. (2026). The avalanche that hit math is coming for AI development. Essay, 13 November 2026. `https://www.dwarkesh.com/p/rsi`

[13] Luo, X., Shinnick, Z., Griesshaber, N., Wang, Y., Yu, J., Shi, F., Torr, P., and Lu, Y. (2026). A language model from 1913: Pretraining on historical text. *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 56150-56165. arXiv:2606.02991.

[14] He, S., Lv, L., Manela, A., and Wu, J. (2025). Chronologically consistent large language models. arXiv:2502.21206.

[15] Turtel, B., Franklin, D., Skotheim, K., Hewitt, L., and Schoenegger, P. (2025). Outcome-based reinforcement learning to predict the future. *Transactions on Machine Learning Research*. arXiv:2505.17989.

[16] Turtel, B., Wilczewski, P., Franklin, D., and Skotheim, K. (2026). Future-as-label: Scalable supervision from real-world outcomes. arXiv:2601.06336.

[17] Baldwin, J. M. (1896). A new factor in evolution. *American Naturalist*, 30(354), 441-451; 30(355), 536-553.

[18] Hinton, G. E., and Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, 1(3), 495-502.

[19] Whitley, D., Gordon, V. S., and Mathias, K. (1994). Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature, PPSN III*, Lecture Notes in Computer Science 866, 5-15. Berlin: Springer. doi:10.1007/3-540-58484-6_245.

[20] Midler, B., and Pan Vazquez, A. (2026). Evolution imposes an inductive bias that alters and accelerates learning dynamics. *eLife*, 15, RP111582 (reviewed preprint). doi:10.7554/eLife.111582.1.

[21] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, PMLR 70, 1126-1135.

[22] Andrychowicz, M., Denil, M., Gómez Colmenarejo, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, 29, 3981-3989.

[23] Metz, L., Harrison, J., Freeman, C. D., Merchant, A., Beyer, L., Bradbury, J., Agrawal, N., Poole, B., Mordatch, I., Roberts, A., and Sohl-Dickstein, J. (2022). VeLO: Training versatile learned optimizers by scaling up. arXiv:2211.09760.

[24] Real, E., Liang, C., So, D. R., and Le, Q. V. (2020). AutoML-Zero: Evolving machine learning algorithms from scratch. *Proceedings of the 37th International Conference on Machine Learning (ICML)*, PMLR 119, 8007-8019.

[25] Miconi, T., Stanley, K. O., and Clune, J. (2018). Differentiable plasticity: Training plastic neural networks with backpropagation. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, PMLR 80, 3559-3568.

[26] Miconi, T., Rawal, A., Clune, J., and Stanley, K. O. (2019). Backpropamine: Training self-modifying neural networks with differentiable neuromodulated plasticity. *International Conference on Learning Representations (ICLR)*.

[27] Najarro, E., and Risi, S. (2020). Meta-learning through Hebbian plasticity in random networks. *Advances in Neural Information Processing Systems*, 33, 20719-20731.

[28] Pedersen, J. W., and Risi, S. (2021). Evolving and merging Hebbian learning rules: Increasing generalization by decreasing the number of rules. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 892-900. doi:10.1145/3449639.3459317.

[29] Wang, F., Tian, H., Xiong, H., Wu, H., Fu, J., Cao, Y., Kang, Y., and Wang, H. (2022). Evolving decomposed plasticity rules for information-bottlenecked meta-learning. *Transactions on Machine Learning Research*. arXiv:2109.03554.

[30] Miconi, T. (2023). Learning to acquire novel cognitive tasks with evolution, plasticity and meta-meta-learning. *Proceedings of the 40th International Conference on Machine Learning (ICML)*, PMLR 202, 24756-24774.

[31] Fernando, C., Sygnowski, J., Osindero, S., Wang, J., Schaul, T., Teplyashin, D., Sprechmann, P., Pritzel, A., and Rusu, A. A. (2018). Meta-learning by the Baldwin effect. *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, 1313-1320. doi:10.1145/3205651.3205763. arXiv:1806.07917.

[32] Oh, J., Farquhar, G., Kemaev, I., Calian, D. A., Hessel, M., Zintgraf, L., Singh, S., van Hasselt, H., and Silver, D. (2025). Discovering state-of-the-art reinforcement learning algorithms. *Nature*, 648, 312-319. doi:10.1038/s41586-025-09761-x.

[33] Ha, D., Dai, A. M., and Le, Q. V. (2017). HyperNetworks. *International Conference on Learning Representations (ICLR)*. arXiv:1609.09106.

[34] Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2), 185-212. doi:10.1162/artl.2009.15.2.15202.

[35] Gaier, A., and Ha, D. (2019). Weight agnostic neural networks. *Advances in Neural Information Processing Systems*, 32.

[36] Hu, N., Mitchell, E., Manning, C. D., and Finn, C. (2023). Meta-learning online adaptation of language models. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4418-4432. doi:10.18653/v1/2023.emnlp-main.268.

[37] Lu, C., Holt, S., Fanconi, C., Chan, A. J., Foerster, J., van der Schaar, M., and Lange, R. T. (2024). Discovering preference optimization algorithms with and for large language models. *Advances in Neural Information Processing Systems*, 37, 86528-86573.

[38] Zweiger, A., Pari, J., Guo, H., Kim, Y., and Agrawal, P. (2025). Self-adapting language models. *Advances in Neural Information Processing Systems*, 38, 74084-74115. arXiv:2506.10943.

[39] Lou, Z., Chen, H., Li, Y., Wang, Q., and Hooi, B. (2026). Meta-TTL: Meta-learning self-improvement policies for language agents. arXiv:2604.00830v4.

[40] Feng, F., Wang, J., Yang, X., and Geng, X. (2025). Learngene: Inheritable 'genes' in intelligent agents. *Artificial Intelligence*, 348, 104421.

[41] Schmidhuber, J. (2003). Gödel machines: Self-referential universal problem solvers making provably optimal self-improvements. arXiv:cs/0309048.

[42] Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., and Balog, M. (2025). AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv:2506.13131.

[43] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. (2017). Population based training of neural networks. arXiv:1711.09846.

[44] Akiba, T., Shing, M., Tang, Y., Sun, Q., and Ha, D. (2025). Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 7, 195-204. doi:10.1038/s42256-024-00975-8.

[45] Zhang, J., Gu, Y., Ruan, J., Song, M., Peng, Y., Han, Z., Xiang, J., Wang, Z., Yang, C., Ouyang, Y., Liu, B., Wu, C., and Luo, Y. (2026). Harnessing agentic evolution. arXiv:2605.13821.

[46] Müller, V., Steels, L., and Szathmáry, E. (2026). Evolvable AI: Threats of a new major transition in evolution. *Proceedings of the National Academy of Sciences*, 123(17), e2527700123. doi:10.1073/pnas.2527700123.

[47] Harris, K. D. (2026). A mathematical theory of evolution for self-designing AIs. arXiv:2604.05142.

[48] Shao, S., Ren, Q., Liu, D., Qian, C., Wei, B., Guo, D., Yang, J., Song, X., Zhang, L., Zhang, W., and Shao, J. (2026). Your agent may misevolve: Emergent risks in self-evolving LLM agents. *International Conference on Learning Representations (ICLR)*, 99728-99793. arXiv:2509.26354.

[49] Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., et al. (2020). The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *Artificial Life*, 26(2), 274-306. doi:10.1162/artl_a_00319.

[50] Halawi, D., Zhang, F., Yueh-Han, C., and Steinhardt, J. (2024). Approaching human-level forecasting with language models. *Advances in Neural Information Processing Systems*, 37. arXiv:2402.18563.

[51] Karger, E., Bastani, H., Yueh-Han, C., Jacobs, Z., Halawi, D., Zhang, F., and Tetlock, P. E. (2025). ForecastBench: A dynamic benchmark of AI forecasting capabilities. *International Conference on Learning Representations (ICLR)*. arXiv:2409.19839.

[52] Zou, A., Xiao, T., Jia, R., Kwon, J., Mazeika, M., Li, R., Song, D., Steinhardt, J., Evans, O., and Hendrycks, D. (2022). Forecasting future world events with neural networks. *Advances in Neural Information Processing Systems*, 35, Datasets and Benchmarks Track.

[53] Goel, S., Chandak, N., Arun, A., Prabhu, A., Staab, S., Hardt, M., Andriushchenko, M., and Geiping, J. (2026). FutureSim: Replaying world events to evaluate adaptive agents. *Advances in Neural Information Processing Systems*, 39, to appear. arXiv:2605.15188.

[54] Lazaridou, A., Kuncoro, A., Gribovskaya, E., Agrawal, D., Liška, A., Terzi, T., Gimenez, M., de Masson d'Autume, C., Kočiský, T., Ruder, S., Yogatama, D., Cao, K., Young, S., and Blunsom, P. (2021). Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34.

[55] Mombaerts, L., Taws, J., Gargano, J. P., Gaddam, D. Y., Tiwari, S., and Rajesh, S. (2026). Training-free continual learning for multimodal forecasting via prompt evolution. *NeurIPS 2026 Workshop on Foundation Models for Temporal Systems: From Forecasting to World Modeling*.

[56] Paleka, D., Goel, S., Geiping, J., and Tramèr, F. (2026). Pitfalls in evaluating language model forecasters. *International Conference on Learning Representations (ICLR)*. arXiv:2506.00723.

[57] Hansen, N. (2016). The CMA evolution strategy: A tutorial. arXiv:1604.00772.

[58] Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R., and Gal, Y. (2024). AI models collapse when trained on recursively generated data. *Nature*, 631(8022), 755-759. doi:10.1038/s41586-024-07566-y.

[59] Gneiting, T., and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477), 359-378. doi:10.1198/016214506000001437.

[60] Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1), 1-3. doi:10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2.

[61] Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting*, 16(4), 437-450.

[62] Perdomo, J. C., Zrnic, T., Mendler-Dünner, C., and Hardt, M. (2020). Performative prediction. *Proceedings of the 37th International Conference on Machine Learning (ICML)*, PMLR 119, 7599-7609.

[63] Sadineni, A. C. (2026). Where an industry keeps its memory: Persistence of regional sector employment after the 2008 and 2020 shocks, with a registered forecast for 2026. Preprint, Zenodo. doi:10.5281/zenodo.24210923.

[64] Bailey, D. H., Borwein, J. M., López de Prado, M., and Zhu, Q. J. (2014). Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society*, 61(5), 458-471. doi:10.1090/noti1105.

[65] Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., and Roth, A. (2015). The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248), 636-638. doi:10.1126/science.aaa9375.

[66] Lichtendahl, K. C., Jr., and Winkler, R. L. (2007). Probability elicitation, scoring rules, and competition among forecasters. *Management Science*, 53(11), 1745-1755. doi:10.1287/mnsc.1070.0729.

[67] Witkowski, J., Freeman, R., Vaughan, J. W., Pennock, D. M., and Krause, A. (2023). Incentive-compatible forecasting competitions. *Management Science*, 69(3), 1354-1374. doi:10.1287/mnsc.2022.4410.

[68] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*.

[69] Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In J.-A. Meyer and S. W. Wilson (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, 222-227. Cambridge, MA: MIT Press.

[70] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, PMLR 70, 2778-2787.

[71] Tonegawa, S. (1983). Somatic generation of antibody diversity. *Nature*, 302(5909), 575-581. doi:10.1038/302575a0.

[72] Burnet, F. M. (1959). *The Clonal Selection Theory of Acquired Immunity*. Nashville: Vanderbilt University Press; Cambridge: Cambridge University Press.

[73] Victora, G. D., and Nussenzweig, M. C. (2012). Germinal centers. *Annual Review of Immunology*, 30, 429-457. doi:10.1146/annurev-immunol-020711-075032.

[74] Eisen, H. N., and Siskind, G. W. (1964). Variations in affinities of antibodies during the immune response. *Biochemistry*, 3(7), 996-1008. doi:10.1021/bi00895a027.

[75] de Castro, L. N., and Von Zuben, F. J. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3), 239-251. doi:10.1109/TEVC.2002.1011539.

[76] Ohno, S. (1970). *Evolution by Gene Duplication*. Berlin: Springer. doi:10.1007/978-3-642-86659-3.

[77] McCloskey, M., and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24, 109-165. doi:10.1016/S0079-7421(08)60536-8.

[78] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13), 3521-3526. doi:10.1073/pnas.1611835114.

[79] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. arXiv:1606.04671.

[80] Carroll, S. B. (2005). Evolution at two levels: On genes and form. *PLoS Biology*, 3(7), e245. doi:10.1371/journal.pbio.0030245.

[81] Kirschner, M., and Gerhart, J. (1998). Evolvability. *Proceedings of the National Academy of Sciences*, 95(15), 8420-8427. doi:10.1073/pnas.95.15.8420.

[82] Dell, M., Carlson, J., Bryan, T., Silcock, E., Arora, A., Shen, Z., D'Amico-Wong, L., Le, Q., Querubin, P., and Heldring, L. (2023). American Stories: A large-scale structured text dataset of historical U.S. newspapers. *Advances in Neural Information Processing Systems*, 36, Datasets and Benchmarks Track, 80744-80772. arXiv:2308.12477.

[83] Hansen, N., and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2), 159-195. doi:10.1162/106365601750190398.

[84] Extavour, C. G., and Akam, M. (2003). Mechanisms of germ cell specification across the metazoans: Epigenesis and preformation. *Development*, 130(24), 5869-5884. doi:10.1242/dev.00804.

[85] Lanfear, R. (2018). Do plants have a segregated germline? *PLoS Biology*, 16(5), e2005439. doi:10.1371/journal.pbio.2005439.

[86] Jablonka, E., and Raz, G. (2009). Transgenerational epigenetic inheritance: Prevalence, mechanisms, and implications for the study of heredity and evolution. *Quarterly Review of Biology*, 84(2), 131-176. doi:10.1086/598822.