

AI Programming Capability Leaps at Fixed Model Scale: Public Evidence and a Testable Hypothesis for ES-DSI as a Societal-Level Mechanism of Intergenerational Evolution

Version: V1.1

Abstract

Current progress in AI capabilities is commonly explained in terms of scaling, improvements in training algorithms, self-training, and inference-time compute. Yet software engineering exhibits a phenomenon that warrants separate study: even without increasing parameter count or architectural capacity—and, in some cases, while retaining the same pretrained base model—post-training can still produce gains of tens of percentage points on software-engineering benchmarks.

In this paper, **fixed model scale** specifically means that parameter count and architectural capacity do not increase. It does not imply that model weights, effective capabilities, post-training compute, or inference strategies remain unchanged. Building on the Knowledge-Structure-Constraint (KSC) framework's distinction between existing knowledge and crystallized structures (K) and new structure generation (S), this paper provides complete definitions and operational accounts of DSI and ES-DSI. **Distributional Structural Internalization (DSI)** describes how the distribution of transferable problem-solving structures in a training environment is translated into a model's initial generative tendencies when it faces new, structurally related tasks. **External Selection-Distributional Structural Internalization (ES-DSI)** describes the intergenerational process through which candidate structures undergo differential retention in the real world, propagate, receive effective exposure in subsequent training, and may thereby alter the initial structural tendencies of later models. The empirical analysis uses progressively controlled, within-family generational comparisons: it controls parameter scale, architecture, base model, and selected training conditions in sequence and observes the capability gains that remain under those controls. The accompanying descriptive dataset currently contains 21 models or versions and 12 within-family longitudinal comparisons.

In a representative case, DeepSeek V4 Flash retained the same architecture and size, and its developer explicitly stated that only post-training was rerun; in DeepSeek's official longitudinal DeepSWE evaluation, its score rose from 7.3 to 54.4, a gain of 47.1 percentage points. GLM-5.3 uses the same base model as GLM-5.2; its official DeepSWE v1.1 score rose from 46.2 to 66.9. As an order-of-magnitude reference, adjacent versions of several frontier models in DataCurve's standardized evaluation—for example, Gemini Flash and Claude Opus—showed DeepSWE gains of approximately 11–18 percentage points. Because longitudinal results across model families do not all come from the same evaluation implementation, and because this paper has not yet defined a “typical frontier-version gain” from a complete sample distribution, these cross-family differences serve only as order-of-magnitude references, not as strict cross-model performance comparisons.

Further controlled studies show that, even when varying degrees of control are imposed on the model, training pipeline, or compute budget, differences in task construction, execution environments, and problem-solving trajectories can still substantially affect generalization to unseen tasks. These results are consistent with DSI's prediction that the distribution of structures in training affects subsequent tendencies in structure generation, but they do not yet directly measure changes in initial structural tendencies. Meanwhile, coding agents already participate in real-world software engineering at scale; observable instances have also emerged for each of the components required by ES-DSI: AI generation, external selection in engineering environments, propagation of digital structures, and subsequent training exposure.

On this evidence, the paper advances the following central mechanistic claim:

ES-DSI should be modeled and tested separately as a societal-level mechanism of AI intergenerational evolution, distinct from scaling, training-algorithm innovation, and closed-loop self-training.

Its basic temporal unit is not a single training run, but “model-real world-subsequent model”: AI systems and humans continually generate candidate problem-solving structures; real environments perform external selection; winning structures change the structural distribution of future digital environments; and subsequent models may convert those changes into new initial structural tendencies through DSI.

Importantly, current public evidence has not yet traced the full causal chain—“generation → selection → propagation → subsequent training exposure → change in initial structural tendencies → performance change”—for the same set of structures. Performance leaps at fixed model scale therefore cannot be directly attributed to ES-DSI. These leaps show only that there is room for capability growth without further increases in model scale, while other evidence separately supports the real-world feasibility of each key ES-DSI component.

These observations make a stronger hypothesis worthy of direct testing:

ES-DSI may already be one of the principal drivers of growth in frontier AI coding and agent capabilities.

The direct evidence in this paper comes primarily from software engineering. Whether this claim generalizes to overall AI capability growth requires independent testing in mathematics, scientific reasoning, and other domains. Existing evidence also does not yet permit a decomposition of the respective contributions of ES-DSI, scaling, training algorithms, and inference-time compute. The claim is therefore explicitly presented as an observable, falsifiable, and quantifiable research hypothesis, not as an empirical conclusion already established by this paper.

Keywords: KSC; DSI; ES-DSI; AI intergenerational evolution; programming capability; post-training; DeepSWE; external selection

1. Introduction: Is There an Underestimated Path of AI Capability Growth?

Over the past several years, scaling has been a major axis for explaining progress in AI capabilities: increasing parameters, data, and training compute has generally improved model performance.

As training systems have grown more complex, reinforcement learning, verifiers, self-training, synthetic data, and test-time compute have also become important sources of capability. These mechanisms are not mutually exclusive, and all continue to develop.

Whether these paths fully account for the principal sources of current AI capability growth, however, remains an open question.

Software engineering is particularly well suited to studying this question. On the one hand, AI programming capability has expanded from local code generation to repository retrieval, cross-file modifications, long-horizon task execution, and interaction with execution environments. On the other hand, code execution, tests, continuous integration, and production feedback subject many candidate solutions to external verification and long-term selection.

This makes a basic question increasingly amenable to empirical testing:

If the parameter count and architecture of the deployed model do not continue to grow, how far can its capabilities still improve?

This paper refers to that condition as “fixed model scale.” The term means only that parameter count and architectural capacity do not continue to increase; it does not imply that post-training, model weights, inference budget, or actual capability remain unchanged. If improvements at fixed model scale are consistently marginal, model scale can still be regarded, to a first approximation, as one of the principal constraints on capability growth. Conversely, if substantial gains continue to occur after fixing parameter count, architecture, and even the base model, then the current picture of AI capability growth must incorporate at least one additional axis beyond scale.

The KSC framework distinguishes among knowledge and crystallized structures available to a task (K), new structure generation (S), and constraint maintenance (C).[1] Within these coordinates, this paper uses DSI and ES-DSI to describe how structures discovered in one period can change the starting point from which later models solve problems. The sections that follow define both mechanisms in full and make their empirical variables and evidentiary boundaries explicit.

This paper addresses three research questions:

RQ1: After fixing parameter count, architecture, or even the base model, can substantial gains in programming capability still occur?

RQ2: When the model, training pipeline, compute budget, or dataset scale is further controlled, do task construction, execution environments, and problem-solving trajectories still systematically affect generalization in ways consistent with DSI's predictions about structural distributions?

RQ3: Once key components such as AI generation, external selection, propagation, subsequent training exposure, and capability change have each become observable, should ES-DSI be modeled as an independent societal-level mechanism of intergenerational evolution, and should we further test whether it has already become an important source of current growth in coding and agent capabilities?

The first two questions provide the empirical foundation. The third contains two distinct levels: proposing ES-DSI as an independent object of study, and advancing the stronger hypothesis that it is “one of the principal drivers.”

2. Analytical Framework: KSC, DSI, and the Complete ES-DSI Mechanism

2.1 The KSC Coordinates Used in This Paper

The Knowledge-Structure-Constraint (KSC) framework distinguishes three functional roles with progressive dependencies in complex goal-directed tasks:[1]

- **K (Knowledge):** information, state, and already-crystallized structures that the system can currently access and use;
- **S (Structure Generation):** the formation, on the basis of K, of new problem representations, variable relations, abstractions, solution schemes, and action paths;
- **C (Constraint Maintenance):** keeping established goals, premises, boundaries, invariants, and acceptance criteria active as constraints on generation and action.

The present paper primarily uses the relationship between K and S. Existing K changes the materials and starting points available to S. New structures produced through S may subsequently be preserved outside the model and, under the conditions specified below, become part of the K available to later models. C is not a principal variable in the intergenerational mechanism proposed here, but it must be distinguished from External Selection: C operates within a task or system by keeping established conditions effective, whereas External Selection acts after candidate structures have been generated and changes their probability of long-term retention.

DSI and ES-DSI further specify how historical structural experience can influence later K and S. To make the argument self-contained, this section fully specifies the unit of analysis, the transmission chain, its observable variables, and the conditions under which the mechanism would or would not produce positive capability accumulation.

2.2 Unit of Analysis: Transferable Problem-Solving Structures

This paper uses **problem-solving structure** to mean a problem representation, variable relationship, procedure, design pattern, solution scheme, search direction, or verification logic that can be reused across tasks that differ in surface details but share relevant underlying relationships.

Merely copying an answer, reproducing a fixed expression, or producing a task-specific output is not sufficient to constitute structural experience in the present sense. What is learned must affect problem solving on new, structurally related instances. In software engineering, examples include ownership and lifecycle relations, patterns for coordinating asynchronous work, rollback structures, test strategies, and ways of decomposing changes across components. The same structure may be expressed through different code, terminology, repositories, or local constraints.

This transfer criterion separates the mechanism from simple repetition. It also defines what later experiments must measure: not merely whether a training example is reproduced, but whether exposure changes which structures are generated early or preferentially on new, related problems.

2.3 DSI: From Structural Experience to Initial Generative Tendencies

This paper defines **Distributional Structural Internalization (DSI)** as:

The process by which training transforms the distribution of transferable problem-solving structures in a data or interaction environment into a model's initial generative tendencies when it faces new, structurally related problems.

The relevant outcome is not simply whether a model can state a concept. It is whether some problem representations, variable relationships, solution patterns, and directions of exploration are more likely than alternatives to enter the initial candidate set before extensive online search, tool feedback, or multi-round revision begins.

For example, when several approaches could be used to manage the lifecycle of an asynchronously loaded shared resource, one model may begin with ad hoc local flags, while another may immediately organize the problem around ownership, reference counting, or an explicit lifecycle protocol. DSI concerns how historical structural experience changes the relative tendency of those candidates to appear and be prioritized.

DSI is not structural determinism. Existing K changes the starting distribution of S ; it does not force the final answer or eliminate the generation of new structures. Nor is DSI a general renaming of the proposition that models learn their training distributions. It is limited to the transfer of problem-solving structures across surface-level variation. Architecture, prompts, decoding settings, and objectives can also affect generative tendencies; a change in such a tendency constitutes evidence for DSI only when it can be linked to a difference in relevant training exposure.

In this paper, **initial structural tendency** is the primary behavioral term. **Structural prior** is used only as shorthand for the same observable asymmetry in early candidate generation; it does not presuppose that a specific Bayesian prior or internal representation has already been identified.

2.4 External Selection: Differential Retention After Generation

AI systems and humans continually produce candidate structures. Generation alone, however, does not determine which structures will shape future environments. Some solutions pass tests, operate successfully, remain maintainable, gain adoption, survive review, and are copied into additional systems; others disappear.

This paper defines **External Selection (ES)** as:

The differential retention of candidate structures, after they are generated, through processes outside the target model's training loop, including engineering execution, tests, production feedback, human judgment, user adoption, peer scrutiny, and long-term practice.

ES does not require a central selector or a single fitness function. It exists whenever different structures acquire different probabilities of survival, reuse, and propagation through real-world processes. It also does not guarantee selection for structural quality: visibility, market incentives, institutional constraints, and network effects can preserve weak structures or suppress valuable ones. Positive accumulation therefore requires external selection that is sufficiently correlated with effectiveness in the relevant domain.

2.5 Propagation and Effective Training Exposure

External selection and DSI are separated by two necessary transmission conditions.

First, selected structures must be **propagated**. Adoption, copying, documentation, reuse in libraries or frameworks, and continued presence in maintained systems change the distribution

of structures in the digital environment. A structure that succeeds once but leaves no persistent trace cannot substantially reweight the environment encountered by later models.

Second, propagated structures must receive **effective training exposure**. Public availability does not imply inclusion in a particular training corpus; inclusion does not imply sufficient weight, diversity of expression, or placement in a training stage that changes later behavior. Model developers can choose data sources and training procedures, but directly constructing training examples with high structural value at sufficient scale remains difficult. Real-world selection and propagation can therefore affect the candidate pool from which later training draws, while training design still determines which part of that pool is actually internalized.

Propagation and effective exposure are transmission conditions, not separate evolutionary mechanisms. If either link fails, the result of ES does not enter DSI.

2.6 The Complete ES-DSI Chain

The complete mechanism can be represented as:

$$G(t) \rightarrow \text{Sel}(t) \rightarrow R(t) \rightarrow E(t+1) \rightarrow D(t+1) \rightarrow K(t+1) \rightarrow S(t+1)$$

where:

- **G**: generation of candidate problem-solving structures by humans, AI systems, or human-AI collaboration;
- **Sel**: differential retention through external selection;
- **R**: propagation of retained structures across artifacts, projects, organizations, and time;
- **E**: effective exposure to those structures in the training or knowledge environments of later systems;
- **D**: the resulting change in initial structural tendencies through DSI;
- **K(t+1)**: the changed knowledge and crystallized structures available to the later model;
- **S(t+1)**: later structure generation beginning from the changed K.

The downstream observable is:

$$D(t+1) \rightarrow \text{DeltaP}(t+1)$$

where **DeltaP** denotes intergenerational change in performance under a specified task distribution, benchmark, agent harness, and inference configuration. DeltaP is not itself part of the transmission mechanism. It is an outcome that may reflect D together with other changes in training algorithms, verifiers, inference-time compute, evaluation conditions, and task composition.

The basic temporal unit of ES-DSI is therefore not one training run, but:

$$\text{model generation} \rightarrow \text{real-world production and selection} \rightarrow \text{subsequent model}$$

A structure that required costly structural search (S) in one period can, after selection, propagation, exposure, and DSI, become a higher-probability initial candidate in the K of a later model. Later systems may then shift from rediscovering that structure toward recognizing and adapting it, freeing S for unresolved parts of the task.

2.7 General Ecological Inheritance and AI Self-Feedback

ES-DSI is not limited to AI-generated structures. Human-generated code, algorithms, theories, designs, and engineering practices can undergo the same sequence and become structural experience for later models. The complete mechanism is therefore first a process of **structural inheritance in an open human-AI ecology**.

The AI-specific feedback component is narrower:

AI or human-AI generation → ES → R → E → DSI in later AI

Attributing later capability change to this feedback requires evidence about the provenance and novelty of the selected structures, their propagation and training exposure, and a relevant counterfactual: without the AI-generated contribution, would later models have acquired the same structure through existing human knowledge? If provenance cannot be separated, the evidence can support general ecological inheritance but cannot by itself estimate the magnitude of AI self-feedback.

2.8 Scope Conditions and Evidentiary Levels

The mechanism does not imply mathematically monotonic capability growth. External selection can be weak or misdirected, effective structures may fail to propagate, training may not expose later models to them, and recursively generated data without adequate selection can degrade model quality.[15]

The argument in this paper therefore separates four evidentiary levels:

- 1. Direct performance observation:** large DeltaP can occur at fixed model scale or with a fixed base model. This establishes room for capability growth without scale expansion, not its cause.
- 2. DSI-consistent controlled evidence:** task construction, execution environments, and interaction trajectories affect generalization under partially controlled training conditions. This narrows plausible explanations but does not directly measure D.
- 3. Component feasibility:** observable instances exist separately for G, Sel, R, and E in software engineering. Separate observations do not establish an end-to-end causal chain.
- 4. Contribution hypothesis:** ES-DSI may already be one of the principal drivers of growth in frontier AI coding and agent capabilities. Establishing that claim requires tracing the same structures through the complete chain and separating the contribution of ES-DSI from those of scaling, training algorithms, verifiers, and inference-time compute.

Within these boundaries, the central mechanistic claim of the paper is that ES-DSI is sufficiently independent, observable, and falsifiable to warrant separate modeling and testing as a societal-level mechanism of AI intergenerational evolution. The stronger claim about its present contribution remains a research hypothesis.

3. Observational Method: Progressively Controlled Within-Family Generational Comparisons

A direct comparison between two model versions does not permit reliable attribution. A single upgrade may simultaneously change:

- total and active parameter counts;
- architecture;
- pretraining data and compute;

- post-training data;
- reinforcement-learning methods;
- verifiers;
- optimizers;
- inference budget.

This paper therefore strengthens the controls in sequence.

L0: Same model family

This reduces confounding from changes in product positioning and target use cases.

L1: Same parameter scale

If significant gains remain, an increase in parameters is not a necessary condition for those gains.

L2: Same architecture

This further excludes architectural replacement as a necessary condition.

L3: Same base model

If capability still improves substantially, retraining a stronger base model is not necessary either.

L4: Fixed model and selected training conditions

Open studies allow further control over the model, training pipeline, training budget, or dataset scale, making it possible to compare the effects of task construction, execution environments, and problem-solving trajectories.

This method is not equivalent to a randomized controlled experiment, nor does it imply strict causal isolation at every level. Under L3 in particular, post-training may still change data, reinforcement learning, verifiers, sampling, and optimization details simultaneously. Its purpose is more limited:

To observe how much performance gain remains after commonly cited capability-growth variables are progressively controlled, and thereby further narrow the range of explanatory variables.

The paper also distinguishes two types of benchmark evidence:

- 1. Longitudinal evaluations from a common source within the same model family**, used to calculate version-to-version differences;
- 2. Independent standardized evaluations such as DataCurve**, used to cross-check final model capability and to provide a common order-of-magnitude reference for changes between adjacent versions of several contemporary frontier models.

Longitudinal differences from different sources are used only to compare orders of magnitude, not for strict cross-source statistical comparison. The current 12 comparisons form a descriptive sample. This paper has not yet specified sufficiently complete inclusion criteria or a statistical distribution to define a “typical frontier-version gain,” and therefore does not use these observations to estimate the population distribution of frontier-model version updates.

4. Evidence I: Large Capability Leaps at Fixed Model Scale

The accompanying descriptive dataset currently contains 21 models or versions and 12 comparisons between adjacent versions within the same family.

The dataset is used here for descriptive comparison, not to estimate a statistical distribution of “typical frontier-model version gains.” A future public release should provide the complete sample, inclusion and exclusion rules, version dates, benchmark versions, agent harnesses, reasoning-effort settings, and other available metadata to improve the reproducibility of longitudinal comparisons.

Unverifiable parameter counts, training methods, and data sources are uniformly marked as **Unknown**. API prices for closed models are used only as auxiliary indicators of product tier and inference cost, not as substitutes for parameter count.

Selected representative results are shown below:

Model transition	DeepSWE gain	Principal controls	Evaluation source
Gemini 3.5 Flash → 3.6 Flash	+11.0 percentage points	Same family, same high setting	DataCurve standardized evaluation [2]
Gemini 3.6 Flash → 3.7 Flash	+18.0 percentage points	Same family, same high setting	DataCurve standardized evaluation [2]
Claude Opus 4.8 → Opus 5	+15.0 percentage points	Same product tier, same max setting	DataCurve standardized evaluation [2]
GLM-5.1 → GLM-5.2	+28.2 percentage points	Same publicly reported parameter scale	Official Z.ai longitudinal evaluation [17]
GLM-5.2 → GLM-5.3	+20.7 percentage points	Same base model	Official Z.ai longitudinal evaluation [5]
DeepSeek V4 Flash Preview → 0731	+47.1 percentage points	Same architecture and size; post-training rerun only	Official DeepSeek longitudinal evaluation [3][4]

Note: The gains across model families do not all come from the same evaluation implementation. Each row is internally comparable over time; differences across rows illustrate only the order of magnitude of the gains and do not constitute a strict cross-model benchmark comparison.

Accordingly, this section describes these gains as “comparable in order of magnitude to changes between adjacent versions of several frontier models,” rather than defining 11-18 percentage points as a statistically “typical frontier-version gain.”

4.1 DeepSeek V4 Flash

DeepSeek's official update notes state that V4-Flash-0731 retained the same model architecture and size as Preview and underwent a new round of post-training.[3]

Its official model card reports the following longitudinal DeepSWE results:

7.3 → 54.4 [4]

Other engineering-related indicators, including Terminal Bench and CyberGym, also improved concurrently.[4]

DataCurve's independent evaluation of the final V4 Flash release was approximately 53%, broadly consistent with the vendor-reported final score of 54.4 and thus providing an independent cross-check of the release model's capability. The Preview score of 7.3, however, comes from the vendor evaluation. The full 47.1-percentage-point longitudinal gain should therefore still be treated as an **official within-vendor longitudinal result**, rather than as an independent benchmark's replication of the entire trajectory.[2][4]

One limitation in the evaluation configuration must also be noted: the public model card states that the 0731 Code Agent evaluation used DeepSeek Harness in minimal mode with **max** reasoning effort, but public materials do not state with equal clarity whether the Preview score of 7.3 was reevaluated under an identical agent execution configuration.[4] This longitudinal result therefore constitutes strong within-vendor evidence, but not yet a strict external replication in which the harness and inference configuration were held constant.

This case supports two relatively strong conclusions about necessary conditions:

A larger parameter scale and a new model architecture were not necessary conditions for this major capability leap.

Public information is insufficient, however, to decompose the post-training change among data, reinforcement learning, reward design, and optimization details.

4.2 GLM-5.2 → GLM-5.3

GLM-5.3 provides a more tightly controlled case in which the base model itself was held fixed.

Z.ai explicitly states that GLM-5.3 uses the same base model as GLM-5.2 and that the improvements come from post-training.[5]

Its official DeepSWE v1.1 result is:

46.2 → 66.9 [5]

Public documentation also indicates that GLM-5.3 continues the core reinforcement-learning strategy already used in 5.2, while emphasizing a broader set of more realistic and more complex software-engineering environments, automated verifiers, solution trajectories, and training tasks.[5]

Therefore:

Retraining a new base model was not necessary to produce an additional software-engineering capability gain of this magnitude.

Both GLM-5.1 and GLM-5.2 are publicly reported as having 744B total parameters and 40B active parameters. Because 5.2 also introduced architectural and training changes, however, this comparison can exclude growth in parameter count as a necessary condition, but cannot exclude architectural change.[17]

5. Evidence II: Training-Data Construction and Interaction-Trajectory Structure Have High Leverage

Evidence from commercial models can narrow the source of change only to the post-training stage. Open research permits further control of the training process.

5.1 SWE-Lego

Using the same Qwen3-32B base model and SWE-bench Verified as its principal software-engineering evaluation, SWE-Lego reports:

- baseline score: 23.2%;
- new software-engineering data: +25.6 percentage points;
- improved supervised fine-tuning: an additional +3.8 percentage points;
- test-time scaling: an additional +6.2 percentage points.[6]

This result cannot be extrapolated to a fixed ratio between data and algorithmic contributions across all frontier models.

What it does support is the following:

On a fixed base model, changes in training experience alone can already produce gains of tens of percentage points in software-engineering capability, without requiring training-method innovation of a comparable magnitude.

5.2 Terminal-Lego

Terminal-Lego further fixes the student model and training pipeline, changing only the problem-solving trajectories used for training.[7]

The study finds that a teacher model's own benchmark score does not directly predict the training value of its trajectories. One important difference lies in the structure of environmental interaction within those trajectories: training trajectories that more frequently exhibit an **inspect -> act -> verify** pattern yield better subsequent performance on terminal tasks.[7]

The value of training data therefore depends not only on whether a task ultimately succeeds, but also on which environmental states, action relationships, and verification processes are exposed during problem solving.

This provides **indirect support** for DSI: different distributions of structures correspond to different subsequent behavior and generalization capabilities, although the study does not directly measure a change in the initial generation probability of any specific transferable structure.

5.3 Dataset Scale Alone Cannot Explain the Results

OpenThoughts-Agent conducts extensive controlled data experiments and, across comparisons involving different training-set sizes under controlled compute, consistently observes performance differences caused by different data-construction methods.[8]

Patches-to-Trajectories reports experiments on tasks including SWE-bench Verified and uses equal-size ablations and compute-matched analyses to show that the quality of intermediate steps in a trajectory cannot be fully explained by trajectory count or additional training compute alone.[9]

Together, these results show:

Training scale and the construction of training data or trajectories cannot be treated as a single variable.

Datasets of the same or similar scale can produce substantially different generalization outcomes when they contain different problem representations, observation-action

relationships, failure paths, and verification processes. Under the definition in Section 2, these results are consistent with DSI's prediction that the distribution of structures in training affects subsequent tendencies in structure generation. This correspondence remains a theoretical mapping made in this paper, however; the existing studies do not directly measure how the initial generation probability of any specific transferable structure changes.

6. Evidence III: Problem-Solving Outputs Involving AI Have Reached a Measurable Scale

ES-DSI first requires earlier-generation AI systems to participate continually in new problem solving and produce digital outputs at sufficient scale.

Software engineering already provides direct proxy evidence for that scale.

Khosravani and Mockus apply multi-signal detection to more than 180 million Git repositories. They find that, by April 2026, commits clearly attributable to coding agents had exceeded:

320,000 per month.

For Claude Code alone, they identify a cumulative total of:

- 886,122 commits;
- 17,295 public projects.[10]

Because much AI-assisted development does not appear under a separate bot identity, these figures are better interpreted as a lower bound on detectable activity.[10]

The AIDev dataset, meanwhile, contains:

932,791 agent-generated pull requests (Agentic PRs) across 116,211 repositories.
[11]

These results directly measure the scale of AI participation in engineering output, not the number of “novel problem-solving structures” within that output.

Accordingly, the more precise conclusion regarding **G**, as defined in Section 2, is:

There is already strong proxy evidence for scale, but structural novelty and structural value have not yet been measured directly.

7. Evidence IV: Real-World Instances of ES, Propagation, and Subsequent Training Exposure

Large-scale generation is only the input condition for ES-DSI. The next question is whether candidate structures undergo differential retention, propagation, and subsequent exposure during training.

7.1 External Selection

Software engineering has multiple layers of real-world selection:

- compilation;
- automated tests;
- continuous integration;

- benchmark evaluation;
- code review;
- human acceptance;
- production operation;
- user feedback;
- defect repair and rollback.

Engineering outputs produced with AI participation therefore do not persist indiscriminately over time.

Ogenrwot and Businge analyze 24,014 merged Agentic PRs, comprising 440,295 commits.[12]

This directly shows:

Some engineering modifications involving AI have passed project-level acceptance and become part of the persistent code history.

Being merged, however, still does not mean that a high-value structure was correctly selected.

Existing evidence therefore strongly supports the **existence of ES**, but its strength, long-term quality, and structural preferences have not yet been systematically quantified.

7.2 Propagation

Structures that survive selection can subsequently enter codebases, tests, APIs, software libraries, frameworks, documentation, workflows, and later projects, thereby propagating across time, projects, and even organizations.

Current public studies mainly quantify agent commits, pull requests, and project adoption. They do not yet systematically trace how often a particular high-value structure is copied, adapted, and propagated over time.

Therefore:

Propagation channels already exist, but R remains a notably weak link in the current evidence chain.

7.3 Subsequent Training Exposure

GLM-5.3's public documentation states that its post-training incorporated an expanded set of new long-horizon software-engineering tasks, execution environments, automated verifiers, and solution trajectories.[5]

Open studies such as SWE-Lego, Terminal-Lego, and OpenThoughts-Agent also directly demonstrate how code repositories, execution environments, and interaction trajectories can be converted into post-training inputs.[6][7][8]

There are therefore direct instances of **new engineering structures being used as inputs in later training**.

One important boundary must nevertheless be preserved:

The presence of AI-generated or AI-assisted structures in public code does not imply that those structures have entered the training set of any particular commercial model.

Thus, E is currently supported by **direct instances of subsequent training exposure**, but the G, Sel, and E observations have not yet been linked in a complete temporal chain over the same structures.

8. Mapping the Evidence to the ES-DSI Chain

Section 2 decomposed the transmission mechanism into G, Sel, R, E, and D, with DeltaP as a downstream performance outcome. The current public evidence maps onto these variables as follows:

Variable	Current software-engineering evidence	Evidentiary status
G	Coding-agent commits and Agentic PRs	Strong proxy evidence for scale; structural novelty not yet directly measured
Sel	Merged PRs, tests, CI, code review, and production feedback	Direct evidence that ES exists; selection strength and quality preferences not yet quantified
R	Propagation through code, software libraries, APIs, tools, and documentation	Channels exist; systematic quantification is lacking
E	New engineering environments, tasks, and trajectories entering post-training	Direct instances exist, but G, Sel, and E have not yet been linked end to end over the same structures
D	Different training trajectories produce different subsequent generalization behavior	Indirect evidence; direct measurement of initial structural tendencies is still lacking
DeltaP	Significant software-engineering performance changes at fixed model scale or with a fixed base model	Strong direct performance evidence

There is still no end-to-end causal record for any frontier commercial model that traces:

G -> Sel -> R -> E -> D

and then tests the causal relationship between D and DeltaP under controlled evaluation conditions. Observable real-world evidence has emerged separately for multiple key components. This supports the existence and feasibility of the components, but the separate observations cannot be combined to establish a closed causal chain.

9. Unclosed Causal Links and Explanatory Boundaries

Publicly available information does not yet permit a reliable decomposition of observed capability gains among scaling, training algorithms, verifiers, inference-time compute, and ES-DSI.

This paper therefore cannot determine what proportion of any specific performance improvement in DeepSeek, GLM, or another model comes from ES-DSI.

Recursive use of model-generated data can itself cause distributional degradation and even model collapse.[15]

Therefore:

Effective external selection is an essential condition for positive accumulation through ES-DSI, not an optional component.

What this paper provides at present is public evidence concerning the existence of the phenomenon, the magnitude of the gains, and mechanistic feasibility—not a complete, end-to-end causal demonstration for a single model.

10. Why the Existing Evidence Shows More Than Merely “Post-Training Works”

That “post-training can make models stronger” is not a new finding of this paper. The current evidence advances the question in three ways.

First, gains at fixed model scale can be large. When parameter count, architecture, and even the base model are controlled to varying degrees, software-engineering benchmark performance can still improve by tens of percentage points, reaching the same order of magnitude as changes between adjacent versions of several contemporary frontier models.

Second, the observed effects concern generalization to unseen tasks. The relevant evaluations include long-horizon software-engineering and terminal tasks, so the issue is not simply memorization or reproduction of training examples.

Third, training experience cannot be characterized by quantity alone. Controlled studies show that, under the same model and similar training budgets, different data-construction methods, patterns of environmental interaction, and trajectory structures can produce substantially different subsequent capabilities.

These observations still **do not directly prove that the performance gains at fixed model scale come from ES-DSI**. They establish a more limited but important empirical foundation: increasing model scale is not necessary for substantial capability growth; the construction of training experience has high leverage; and the generation, selection, propagation, and subsequent exposure required by ES-DSI have each emerged separately in the real world. This moves ES-DSI from a purely theoretical possibility to a mechanistic hypothesis that can be tested by tracking the same structures end to end and decomposing the respective contributions of different mechanisms.

11. Discussion: ES-DSI as a Societal-Level Mechanism of AI Intergenerational Evolution

Section 2 defined the mechanism in full; Sections 4–10 established the current evidentiary basis and its limits. This discussion develops the paper's central mechanistic claim:

ES-DSI should be modeled and tested separately as a societal-level mechanism of AI intergenerational evolution, distinct from scaling, training-algorithm innovation, and closed-loop self-training.

Here, **societal-level** means that structure generation, selection, and propagation are distributed across large numbers of independent human organizations, real-world tasks, and digital systems, rather than being confined to the internal loop of a single model-training process.

11.1 Relation to Prior Work on Intergenerational Accumulation

Prior research has already examined intergenerational accumulation in AI. Cook et al. propose Artificial Generational Intelligence, studying individual exploration, social learning, and intergenerational information transmission in controlled reinforcement-learning environments, and demonstrating the possibility of cultural accumulation in artificial agents.[13]

Closer to the present question, Ferbach et al. study models that recursively consume their own generated data (self-consuming generative models). They analyze how data selected by users or reward models enters iterative retraining and prove that, under their theoretical setting, data curation can produce implicit preference optimization while potentially amplifying biases in the selection function itself.[18]

This paper therefore does not claim to be the first to propose that “AI can exhibit intergenerational accumulation” or that “selected model-generated data can affect later models.”

ES-DSI differs in its focus:

It focuses on structure generation by AI systems and humans in open, real-world production environments; real-world external selection independent of model-training objectives; propagation of digital structures across projects and organizations; and renewed exposure to those structures during the training of subsequent general-purpose models.

In other words, ES-DSI is not a closed iterative retraining protocol. It seeks to describe how real-world digital society becomes an environment for structural selection and accumulation across model generations.

11.2 Society as a Whole Pays the Cost of Structural Search and Selection

New problem-solving structures are not produced only by model laboratories.

Across society, people and organizations continually engage in software development, scientific research, engineering design, product decisions, data analysis, and the creation of new tools and workflows. These activities inherently require continued search for new problem representations, variable relationships, solution paths, and verification methods.

That is:

Real-world productive activity continuously pays the cost of structural search.

As AI systems participate more deeply in these activities, a growing share of the search results is preserved in digital forms such as code, papers, tests, designs, and tools.

When winning structures enter future training environments, subsequent models may make use of:

problem-solving structures produced after society has borne the cost of search, and then retained through real-world selection.

This gives ES-DSI a cost structure markedly different from conducting additional training experiments specifically to improve a model.

It does not imply that all structures are publicly shared. Private code, closed data, and internal organizational experience continue to exist. More precisely:

Structure generation and selection occur across digital society as a whole and have substantial potential for cross-organizational spillover.

11.3 Difference from Scaling: Expanding Model Scale Versus Accumulating Structural Experience

Scaling advances model capabilities by increasing parameters, data, and training compute.[16]

This path remains important, but increasing training scale requires additional resources, while larger deployed models generally place greater demands on memory, communication, or inference compute, although the magnitude varies.

ES-DSI focuses on a different variable:

Under fixed model scale, which externally selected problem-solving structures are repeatedly exposed in future training environments and internalized by subsequent models?

The distinction can therefore be stated approximately as follows: scaling changes the representational and optimization resources provided by parameters, data, and training compute; ES-DSI seeks to explain how intergenerational structural experience changes the learning distribution and initial structural tendencies at a given model scale.

The fixed-scale capability leaps observed in this paper show only that capability growth does not require a simultaneous increase in deployed model scale. They do not by themselves prove that the gains come from ES-DSI.

Nor does this mean that ES-DSI has no training cost, or that the total inference cost of real agents must remain unchanged. More precisely:

The capability-growth path described by ES-DSI does not require deployed model scale to increase in step with capability gains.

If the mechanism's contribution is ultimately validated, it would provide an account of how capability growth can be partially decoupled from deployed model scale.

11.4 Difference from Self-Training: External Selection

Self-training, synthetic data, and self-play can already form effective learning loops in some domains with reliable verifiers.

For example, Self-Rewarding Language Models allow models to generate their own training examples and use the model itself as a judge in iterative optimization.[14] Ferbach et al. further show that generated data subjected to external curation can alter the distributions of subsequent models through iterative retraining.[18]

What distinguishes ES-DSI from both self-training and curated retraining is therefore not whether AI-generated data is used.

The key distinction is:

Do the primary selection signals determining which structures prevail over time come from outside the model's own training loop?

Typical self-training can be approximated as:

model generation -> internal reward / verifier -> retraining

ES-DSI additionally includes:

model and human generation -> real-world external selection -> propagation -> subsequent training

An engineering structure's persistence depends not only on whether a training system judges it correct, but also on real-world operation, human maintenance, user adoption, failures and rollbacks, and continued reuse by other projects.

Therefore:

The key to ES-DSI is not AI self-replication, but intergenerational inheritance after independent real-world selection.

11.5 Difference from Training-Algorithm Innovation: Discrete Breakthroughs Versus Continuous Accumulation

New architectures, optimizers, reinforcement-learning algorithms, and inference methods usually require method design, training experiments, failed explorations, engineering implementation, and validation at scale.

This form of capability growth typically takes the form of discrete innovation; neither its timing nor the magnitude of its returns can be predicted in advance.

The inputs to ES-DSI, by contrast, come from ongoing real-world problem solving. New inheritable structures can continue to emerge as long as AI continues to participate in new real-world tasks, external selection remains effective, winning structures can be preserved and propagated, and subsequent models encounter them again.

Training-algorithm progress therefore tends to appear as changes in methods, implementations, or training configurations; in theory, ES-DSI allows real-world problem solving to provide a continuous stream of new inheritable structures. The two are not mutually exclusive, and their respective contributions cannot be apportioned on the basis of temporal continuity alone.

11.6 Directional Accumulation and Self-Reinforcement

ES-DSI does not guarantee mathematically monotonic increases in model capability. Selection can fail, and erroneous structures can propagate.

But if real-world external selection generally raises the persistence probability of effective structures, future digital environments will be continually reweighted toward such structures.

This produces the following feedback loop:

higher K -> stronger generation capability -> more complex new problems solved -> more candidate structures -> another round of selection

Calling this process “automatic” does not mean that no humans participate. Rather:

No single actor needs to pursue AI evolution as an explicit goal: firms, researchers, developers, and users solve problems for their own real-world objectives, yet the evolutionary process can continue as a by-product of societal production.

Under conditions of persistently effective ES, ES-DSI therefore tends toward directional accumulation and self-reinforcement.

11.7 Comparison with Other Capability-Growth Mechanisms

Mechanism	Primary level of operation	Principal new resource	Requires expanded model capacity?	Inference-resource requirement	Primary selection signal	Temporal character
Scaling	Single model generation	Parameters, training compute, data	Commonly associated	Usually increases resource pressure	Training objective	Scale expansion, typically with diminishing marginal returns
Training-algorithm innovation	Single-generation training	New algorithms, experiments, and engineering	No	Not necessarily	Human-designed objectives / verifiers	Discrete innovation
Self-training	Single-generation or recursive training	Model-generated data	No	Not necessarily	Feedback internal to the model or training system	Depends on closed-loop quality
ES-DSI	Across model generations and real-world society	Structures generated and selected through real-world problem solving	No	Does not require model capacity to expand in step with capability gains	Engineering systems, humans, users, and real environments	Continuous, cross-organizational, directionally cumulative

These mechanisms are not mutually exclusive. Contemporary frontier AI likely gains capabilities from several of them simultaneously.

The distinctive feature of ES-DSI is:

The actors performing structural search and selection extend beyond model laboratories.

11.8 Core Hypothesis: ES-DSI May Already Be One of the Principal Drivers of Capability Growth

The preceding evidence supports three observations. First, large software-engineering capability gains can occur even at fixed model scale or with a fixed base model. Second, training-data construction, execution environments, and interaction trajectories have substantial effects in controlled studies. Third, observable instances have emerged separately for AI structure generation, real-world external selection, digital propagation, and subsequent training exposure.

The evidence does not yet close the $G \rightarrow \text{Sel} \rightarrow R \rightarrow E \rightarrow D$ transmission chain over the same structures, nor does it establish a causal relationship between D and DeltaP. It therefore cannot directly establish that ES-DSI explains the observed fixed-scale performance gains, much less that its contribution already exceeds those of other mechanisms.

Together, these observations provide a real-world basis for directly testing the core hypothesis advanced in this paper:

ES-DSI may not be merely a secondary mechanism whose importance lies in the future; it may already be one of the principal drivers of growth in frontier AI coding and agent capabilities.

This is the paper's core hypothesis; the available public evidence is not yet sufficient to establish it.

The direct empirical foundation is currently concentrated in software engineering. A further extrapolation, not directly supported by this paper, is:

ES-DSI may also become an important driver of overall frontier AI capability growth.

That broader claim requires independent replication in mathematics, scientific reasoning, open-ended research, and other domains.

A strict test of the “principal driver” hypothesis ultimately requires comparing:

Contribution_ES-DSI

with:

Contribution_Scaling
Contribution_TrainingAlgorithm
Contribution_Inference

If future research finds that ES-DSI explains only a very small share of capability change after other variables are controlled, the hypothesis should be weakened. Conversely, if end-to-end tracking shows that structures propagated after ES and encountered during training consistently cause measurable changes in initial structural tendencies, and these changes further explain a large proportion of performance gains at fixed model scale, then ES-DSI should be elevated to a core variable for understanding AI capability progress.

12. Falsifiable Predictions and a Quantification Roadmap

The research value of ES-DSI is that its individual links can be measured separately and then tested together as a complete temporal chain.

12.1 G: Provenance and Structural Novelty

Select candidate structures whose origin can be observed prospectively. Record:

- whether they were generated by humans, AI systems, or human-AI collaboration;
- whether they were genuinely new to the relevant environment or restatements of established structures;
- which components of the structure required new search rather than retrieval;
- the counterfactual likelihood that the same structure would have appeared without the AI contribution.

Raw output volume is not sufficient. If little new problem-solving structure can be identified in AI-generated artifacts, those artifacts can show the scale of AI participation but cannot support a reliable estimate of AI-specific intergenerational feedback.

12.2 Sel: Contribution of External Selection

Hold the model, training algorithm, dataset scale, and training compute fixed, and compare:

AI-generated structures not subjected to real-world selection

with:

structures selected through testing, human screening, real execution, or production feedback.

If the external-selection effect proposed by ES-DSI holds, the latter should produce more stable cross-task generalization gains.

The comparison should also distinguish among different sources of selection—tests, production survival, human review, user adoption, and model-internal rewards—because ES-DSI predicts that the source and quality of selection affect which structures persist.

12.3 R: Structural Propagation Multiplier

Track structures that have passed through ES:

- how many times they are copied;
- how many projects and organizations they enter;
- whether they enter software libraries, frameworks, documentation, or standard implementations;
- whether they give rise to derivative versions;
- how long they survive.

These measures would estimate how strongly selected structures reweight subsequent digital environments. A structure can succeed locally while making almost no contribution to intergenerational inheritance if its propagation multiplier remains low.

12.4 E: Effective Training Exposure

For propagated structures, further measure:

- whether they enter subsequent training or knowledge environments;
- the number and timing of exposures;
- the diversity of forms in which they appear;
- their weight relative to competing structures;
- the training stage and objective under which they are encountered.

E is the bridge through which changes in real-world structural distributions can enter DSI. Public availability alone is not evidence of effective exposure for a particular model.

12.5 D: Change in Initial Structural Tendencies

Select clearly defined problem-solving structures and control their:

- exposure frequency;
- diversity of expression;
- contexts;
- training stage.

Then, on new tasks, measure:

the probability that the structure is generated naturally or invoked preferentially.

Measurements should be taken before extensive tool use, external feedback, or multi-round correction so that they capture the starting distribution of S rather than eventual task success. This would advance DSI from benchmark-mediated inference to direct behavioral measurement of initial structural tendencies.

12.6 Full-Chain Tracing and Contribution Decomposition

The strongest research design would select a set of problem-solving structures that are genuinely new at the time of observation and track them continuously through:

```
G: initial generation and provenance
-> Sel
-> R
-> E
-> D
```

The design should then test whether the measured D produces a downstream performance change:

```
D -> DeltaP
```

This design should compare later models or controlled training runs with and without exposure to the tracked structures while holding model scale, algorithms, verifiers, inference configuration, and evaluation conditions as constant as possible. Only at this stage can the contributions of ES-DSI, scaling, training algorithms, and inference-time compute begin to be compared rigorously.

The principal-driver hypothesis would be weakened if end-to-end tracing repeatedly found that externally selected and propagated structures rarely entered later training, failed to alter initial structural tendencies, or explained only a small residual share of DeltaP after competing

variables were controlled. Conversely, the hypothesis would receive stronger support if the same structures could be traced through the complete chain and accounted for a substantial share of fixed-scale capability growth.

13. Brief Industry Implications

If ES-DSI is ultimately shown to make an important contribution to current growth in coding and agent capabilities, the key assets of model companies cannot be measured solely by training compute and parameter scale.

Networks of real-world tasks capable of continually generating, selecting, propagating, and reintegrating high-value structures into training may also constitute important infrastructure for capability growth.

Significant gains at fixed model scale further suggest that, once models enter a certain scale range, the optimal allocation of resources between continued expansion and improvements to training data, environments, and structural experience deserves study as an independent question.

14. Conclusion

This paper begins with an empirical phenomenon: even when parameter scale and model architecture—and, in some cases, the base model—are held fixed, software-engineering capability can still improve by tens of percentage points on benchmarks, reaching the same order of magnitude as changes between adjacent versions of several contemporary frontier models.

Using the K-S relationship in the KSC framework, the paper specifies the unit of analysis and the full intergenerational mechanism required to analyze this phenomenon. Transferable problem-solving structures generated in one period may undergo external selection, propagate through digital environments, receive effective exposure in later training, and alter the initial structural tendencies of subsequent models through DSI. Analytically, this transmission mechanism should be distinguished from its downstream performance outcome.

Progressively controlled model comparisons show that parameter and architectural expansion are not necessary conditions for these gains. Controlled studies further show that training-data construction, execution environments, and interaction trajectories substantially affect later generalization. Observable instances have also emerged separately for AI generation, external selection, digital propagation, and subsequent training exposure in the real-world software-engineering ecosystem.

The evidence has not yet connected these components into a complete causal chain over the same structures for any specific frontier model. This paper therefore does not directly attribute the fixed-scale performance leaps of DeepSeek, GLM, or other models to ES-DSI.

Within that boundary, the paper proposes:

ES-DSI should be modeled and tested separately as a societal-level mechanism of AI intergenerational evolution, distinct from scaling alone, training-algorithm innovation, and closed-loop self-training.

The key mechanism is not the unconditional relearning of AI's own outputs. Rather, candidate structures generated during real-world problem solving undergo independent external selection,

alter the structural distribution of future digital environments, and re-enter subsequent training, thereby increasing their probability of becoming initial candidates for the next generation of models.

The current public evidence further makes a stronger hypothesis worthy of direct testing:

ES-DSI may already be one of the principal drivers of growth in frontier AI coding and agent capabilities.

The causal chain underlying this claim has not yet been closed, nor have the respective contribution shares been quantified. The claim therefore cannot be directly extrapolated into an established fact about overall AI capability growth.

The next stage of research should therefore be divided into two sequential questions:

- 1. End-to-end causal tracing:** For genuinely new structures, can continuous observation establish the temporal and quantitative relationships among $G \rightarrow \text{Sel} \rightarrow R \rightarrow E \rightarrow D$, and can it then test the causal relationship between D and ΔP ?
- 2. Contribution decomposition:** After controlling for scaling, training algorithms, verifiers, and inference-time compute, how much capability gain does ES-DSI actually explain?

Only after the first step is completed can the second question—“How much has ES-DSI already contributed?”—acquire strict causal meaning. If this chain can be measured reliably, AI intergenerational evolution can move beyond a macro-level narrative and become an axis of capability growth whose contribution can be analyzed alongside scaling, training algorithms, and inference-time compute.

References

- [1] Liu, M. **KSC: Knowledge, Structure, and Constraint—A Functional Framework for Human-AI Differences and Collaboration**. Zenodo, 2026. DOI: [10.5281/zenodo.21834579](https://doi.org/10.5281/zenodo.21834579).
- [2] Huang, W., & Jiang, P. **DeepSWE v1.1: A Cleaner, More Reproducible Benchmark for Frontier Coding Agents**. DataCurve, 2026.
- [3] DeepSeek. **DeepSeek API Change Log: DeepSeek-V4-Flash Update**. 2026.
- [4] DeepSeek-AI. **DeepSeek-V4-Flash-0731 Model Card**. Hugging Face, 2026.
- [5] Z.ai. **GLM-5.3: Frontier Coding with Emergent Cyber Capabilities**. 2026.
- [6] Tao, C., Chen, J., Jiang, Y., et al. **SWE-Lego: Pushing the Limits of Supervised Fine-tuning for Software Issue Resolving**. arXiv:2601.01426, 2026.
- [7] Yang, S., Tao, C., Chen, J., et al. **What Makes Interaction Trajectories Effective for Training Terminal Agents?** arXiv:2606.03461, 2026.
- [8] Raoof, N., Zhuang, R., Nezhurina, M., et al. **OpenThoughts-Agent: Data Recipes for Agentic Models**. arXiv:2606.24855, 2026.
- [9] Ma, M., Chen, T., Lin, Y., et al. **From Patches to Trajectories: Privileged Process Supervision for Software-Engineering Agents**. arXiv:2605.21996, 2026.
- [10] Khosravani, A., & Mockus, A. **Detecting AI Coding Agents in Open Source: A Validated Multi-Method Census of 180 Million Repositories**. arXiv:2606.24429, 2026.
- [11] Li, H., Zhang, H., & Hassan, A. E. **AIDev: Studying AI Coding Agents on GitHub**. arXiv:2602.09185, 2026.

- [12] Ogenrwot, D., & Businge, J. **How AI Coding Agents Modify Code: A Large-Scale Study of GitHub Pull Requests**. arXiv:2601.17581, 2026.
- [13] Cook, J., Lu, C., Hughes, E., Leibo, J. Z., & Foerster, J. **Artificial Generational Intelligence: Cultural Accumulation in Reinforcement Learning**. arXiv:2406.00392, 2024.
- [14] Yuan, W., Pang, R. Y., Cho, K., et al. **Self-Rewarding Language Models**. arXiv:2401.10020, 2024.
- [15] Shumailov, I., Shumaylov, Z., Zhao, Y., et al. **AI Models Collapse When Trained on Recursively Generated Data**. Nature, 631, 755-759, 2024.
- [16] Kaplan, J., McCandlish, S., Henighan, T., et al. **Scaling Laws for Neural Language Models**. arXiv:2001.08361, 2020.
- [17] Z.ai. **GLM-5 Official Repository and Model Specifications**. 2026.
- [18] Ferbach, D., Bertrand, Q., Bose, A. J., & Gidel, G. **Self-Consuming Generative Models with Curated Data Provably Optimize Human Preferences**. Advances in Neural Information Processing Systems, 37, 2024. arXiv:2407.09499.