Preprints.org

Article

Not peer-reviewed version

Data-Centric Recursive Improvement for Foundation Models: A Survey

Yanyan Zou [*,†,‡], Chao Chen [‡], Tung Sum Thomas Kwok [‡], Jiayu Yang [‡], Yunyun Hou [‡], Jingmin Zhu [‡], Zehang Luo [‡], Shengjie Li [‡], Yinhong Liu [‡], Yingjia Wan [‡], Chengzu Li [‡], Tao Yu, Zhijiang Guo [*], Jiachen Liu [*], Sirui Wang, Nan Duan

Posted Date: 2 September 2026

doi: 10.20944/preprints202609.0154.v1

Keywords: data-centric artificial intelligence; recursive self improvement; data-evalution co-evolution

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC, OpenAlex.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Data-Centric Recursive Improvement for Foundation Models: A Survey

Yanyan Zou [1,*,†,‡], Chao Chen [2,‡], Tung Sum Thomas Kwok [3,‡], Jiayu Yang [2,‡], Yunyun Hou [4,‡], Jingmin Zhu [5,‡], Zehang Luo [2,‡], Shengjie Li [6,‡], Yinhong Liu [7,‡], Yingjia Wan [3,‡], Chengzu Li [7,‡], Tao Yu [8], Zhijiang Guo [2,*], Jiachen Liu [1,*], Sirui Wang [1] and Nan Duan [1]

[1] JD Future Academy
[2] Hong Kong University of Science and Technology (Guangzhou)
[3] University of California, Los Angeles
[4] Communication University of China
[5] Monash University
[6] Peking University
[7] University of Cambridge
[8] Institute of Automation, Chinese Academy of Sciences
* Correspondence: zoe.yyzou@gmail.com (Y.Z.); zhijiangguo@hkust-gz.edu.cn (Z.G.); liujiachen.jason@jd.com (J.L.)
† Project lead.
‡ Core contributor.

Abstract

Data-centric methods are central to foundation model development, but data construction and evaluation have often been treated as separate stages. Recent model-development pipelines increasingly couple the two: evaluation results expose capability gaps and failure modes, which then guide data selection, filtering, synthesis, and post-training interventions that shape the behavior of subsequent model versions. This survey studies this trend as data-centric recursive improvement, with data-evaluation co-evolution as its core mechanism. In this mechanism, evaluation produces diagnostic signals, orchestration determines whether and how these signals should be trusted and used, and execution updates mutable data-related objects. These updated objects are then used in later training, adaptation, retrieval, or memory stages, enabling the system to change in subsequent rounds. The framework centers on three questions: *what signal diagnoses the current system, who decides how to act on that signal, and what object is updated to affect the next round*. We organize the literature around this signal-decision-update loop, covering feedback signals, control decisions, and data updates across major stages of foundation model development. We further examine failure modes in closed-loop improvement, such as unreliable feedback, feedback overfitting, unstable updates, erosion of data support, and irreproducible system changes. By providing a structured framework for understanding and designing data-evaluation co-evolution, this survey outlines a roadmap toward more adaptive, robust, and accountable foundation model development, where evaluation becomes an active driver of recursive model improvement rather than a passive measure of progress and ultimately sheds light on possible pathways toward Artificial Super Intelligence (ASI), in which a model continually evolve and surpass human-level performance across diverse tasks.

Keywords: data-centric artificial intelligence; recursive self improvement; data-evalution co-evolution

1. Introduction

Foundation-model development is increasingly organized around repeated cycles of diagnosis and repair rather than a single pass of dataset construction followed by post-hoc evaluation. In current practice, an evaluation result is often not the end of an experiment but the beginning of the next data intervention: benchmark failures [1] or learned judges [2,3] motivate targeted data acquisition. These examples point to a broader shift: evaluation is becoming an operational part of the improvement

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

process, and data is no longer a static input but a mutable state shaped by what the current model fails to do.

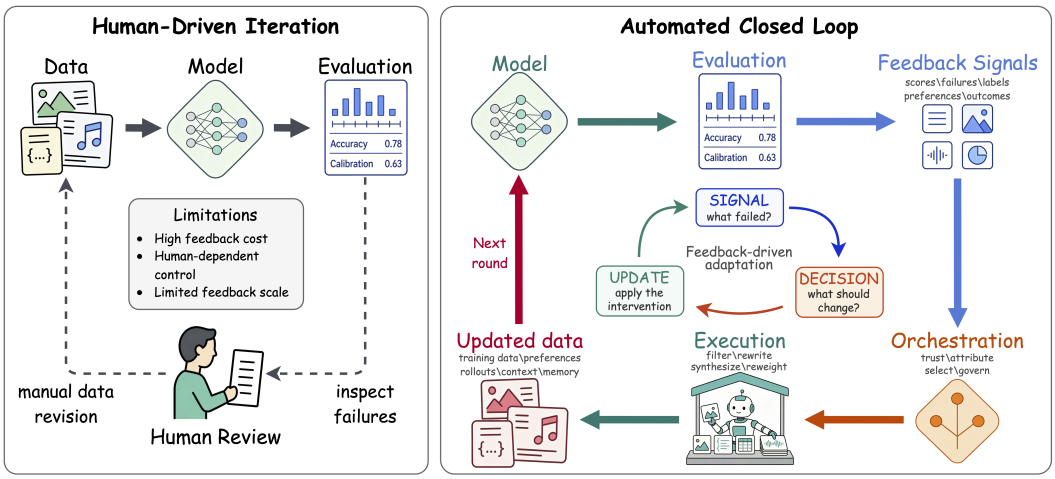


Figure 1. From human-driven iteration to the automated closed loop we formalize as *data–evaluation co-evolution*. Foundation-model development has traditionally followed a linear pipeline (left) in which humans manually inspect evaluation outputs and revise data for the next round. This survey studies the automated regime (right), in which evaluation signals grounded in text, image, audio, or preference feedback close the loop through orchestration and execution over mutable data-related state.

This shift is only partially captured by existing views of data-centric AI, evaluation, and model self-improvement. Data-centric work has provided strong tools for documenting datasets [4], tracking provenance [5], filtering web corpora [6], and designing data mixtures [7,8], but it often studies data quality as a property of a dataset or pipeline before the next model is trained. Evaluation surveys summarize benchmark design, contamination-aware evaluation, and the shift from static to dynamic evaluation [9,10], but this line of work often treats evaluation as a measurement layer rather than as a controller of subsequent data changes. Self-improvement and agent-evolution surveys cover broad mechanisms by which models, agents, tools, or memories adapt [11], yet they typically do not isolate the data pathway through which feedback changes later training. As a result, methods such as dynamic data construction [3], target-aware instruction selection [12], and failure-driven tuning [13] are discussed as different topics even when they share the same underlying loop. The central lens of this survey is therefore the operational path from evaluation signal to data-centric update. Across these systems, three operational questions recur: what signal diagnoses the current model state, who or what determines whether that signal is trustworthy and translates it into an intervention, and what object is updated to shape future model behavior?

Our scope is intentionally data-centric: we focus on improvement driven by data-related objects. We introduce *data-centric recursive improvement* to describe improvement processes in which a foundation-model system becomes better across rounds by updating such objects. To the best of our knowledge, this is the first systematic survey centered on data-centric recursive improvement, offering a foundational roadmap for understanding how evaluation can drive model improvement through data flows.

Since this line of research is still taking shape, its scope and terminology have not yet converged on a single agreed-upon formulation. We therefore treat this survey not as a catalogue of a settled paradigm, but as an effort to clarify an emerging set of related mechanisms. Rather than drawing overly strict boundaries, we use a unified framework to connect the diverse approaches that are beginning to appear across the literature. Specifically, we define the core mechanism as *data-evaluation co-evolution* and structure our analysis around the three foundational questions introduced above through three connected elements: evaluation, orchestration, and execution. An evaluation signal diagnoses the current system; an orchestration policy decides whether the signal is trustworthy, what

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

failure it indicates, and which intervention it justifies; and an execution mechanism applies the selected intervention to a mutable data-related state that affects a later round. This separation matches the functional roles exposed by recent systems. DataOrchestra [14] separates tool-execution feedback from chunk-level curation decisions. BigBang [15] uses critic judgments and post-training outcomes to revise later synthesis and evaluation policies. Frontis-MA1 [16] updates programs and experience records rather than model weights during inference. A method falls within our core scope when this signal-to-decision-to-update path is explicit enough to show how evaluation changes a later data object, context, memory, or control policy. Under this view, an aggregate score, failure cluster, verifier label, preference comparison, judge rationale, or environment outcome becomes more than a report only when it is consumed by a decision process that changes future training or adaptation.

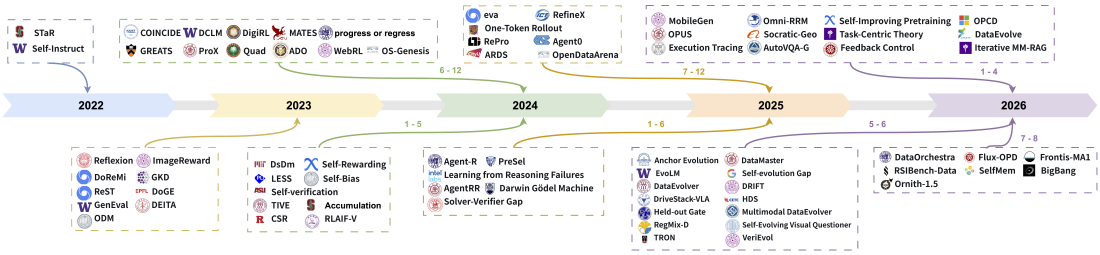


Figure 2. An evolutionary landscape of several representative data-evolution co-evolution frameworks from 2022 to 2026.

2. Definition and Taxonomy

2.1. Definition

We define *data–evaluation co-evolution* as a feedback loop in which evaluation signals guide updates to data-related objects, and those updated objects affect later model or system behavior that is evaluated again in later rounds.

This concept is our mechanism-level view of *data-centric recursive improvement*. Recursive improvement is the broader phenomenon: a system state produced in one round contributes to later improvement. Data-evaluation co-evolution specifies how this happens through data-centric updates. Evaluation diagnoses the current system, orchestration decides how the signal should be used, and execution changes the data-related object that affects future behavior. These roles answer three questions that define the loop: what has been observed, how the observation should be used, and what object is actually changed.

2.2. Data-Evaluation Co-Evolution as the Improvement Mechanism

We formalize data-evaluation co-evolution as a closed loop with three functional roles: evaluation, orchestration, and execution. Evaluation observes the current model or system state and produces an actionable signal. Orchestration interprets this signal and selects an intervention. Execution applies the intervention to a mutable data-related object. The updated state is then evaluated again, making the next round of improvement depend on the outcome of the previous round. An instance loop is depicted in Figure 3.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

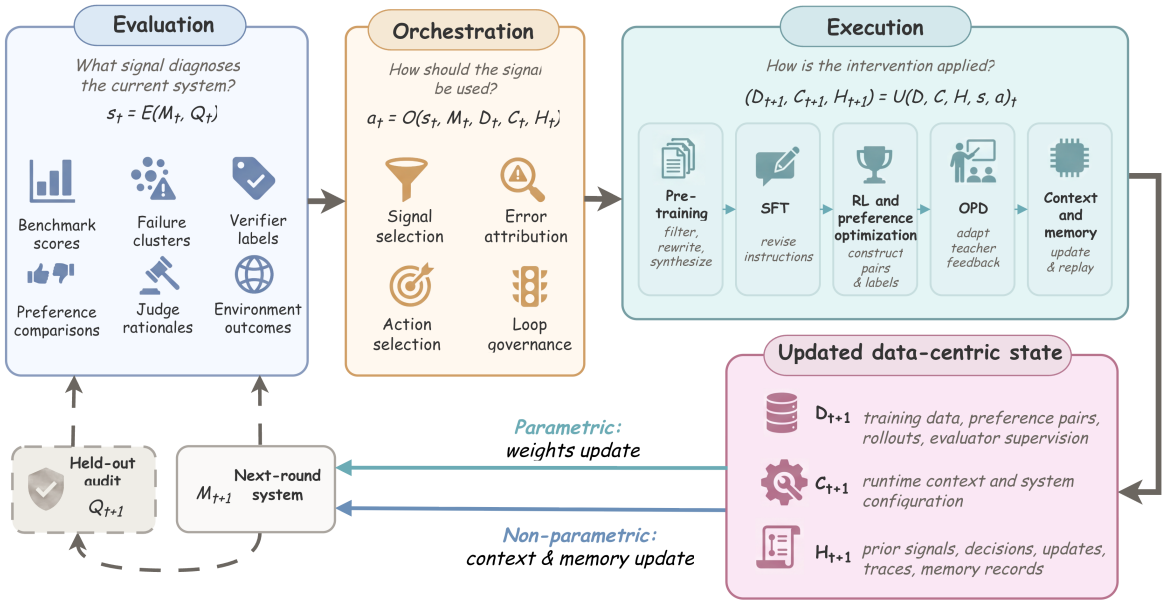


Figure 3. The illustrated signal-to-decision-to-update path shows how evaluation changes a later data-related state. Data–evaluation co-evolution is a state transition rather than a measurement step. Evaluation observes the current system and returns a signal, orchestration decides whether that signal is trusted and which change it justifies, and execution applies the selected action to a mutable data-related object that the next round inherits. The updated object reaches the next system through a parameter-updating path or a non-parametric path. A held-out audit distribution stays outside the update operator, so a gain measured by the controller's own evidence does not by itself establish improvement. Equations (1) to (3) formalize the three roles.

Formally, let $\mathcal{E}$ denote the evaluation procedure that produces diagnostic evidence from the current system. It may be implemented by a benchmark or a verifier model; see Section 3 for more details. We denote by M_t the model or model-based system at iteration t. $\mathcal{Q}_t$ denotes the evaluation distribution at iteration t, such as benchmark examples, tool checks, or environment states used to evaluate the current model or model-based system. It may be fixed or refreshed across rounds.

First, an evaluation function maps the current system and an evaluation distribution $\mathcal{Q}_t$ to a feedback signal:

$$s_t = \mathcal{E}(M_t, \mathcal{Q}_t). \tag{1}$$

The signal s_t may summarize aggregate performance or isolate a specific failure. Depending on the setting, it may take the form of human or model feedback, verifier or tool outputs, or external environment signals, as discussed in Section 3.1. A signal becomes relevant to recursive improvement only when it can guide a later update, rather than merely report performance.

Second, the orchestration function $\mathcal{O}$ interprets the actionable signal into a specific intervention:

$$a_t = \mathcal{O}(s_t, M_t, \mathcal{D}_t, C_t, \mathcal{H}_t), \tag{2}$$

where $\mathcal{O}$ denotes an orchestration policy, such as a judge, a planner, or a scheduler, as discussed in Section 4. $\mathcal{D}_t$ denotes the mutable data-related object available to the system at iteration t, for example, training data, preference pairs, rollout traces, and evaluator-supervision data. C_t denotes runtime context or system configuration, and $\mathcal{H}_t$ denotes the loop history, including previous signals, decisions, updates, traces, and memory records when available. This step decides whether the signal should be trusted, what failure it indicates, and which intervention should follow.

Third, the execution function applies the selected intervention:

$$(\mathcal{D}_{t+1}, C_{t+1}, \mathcal{H}_{t+1}) = \mathcal{U}(\mathcal{D}_t, C_t, \mathcal{H}_t, s_t, a_t), \tag{3}$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

where $\mathcal{U}$ denotes an update operator, such as data filtering, reweighting, rewriting and synthesizing, as formalized in Section 5. The updated memory $\mathcal{H}_{t+1}$ records the interaction trajectory and supports tracing previous loop states. After the intervention is applied, the updated data, context, and memory serve as inputs to later training or adaptation, resulting M_{t+1} that denotes a parameter-updated model or a model-based system consuming $\mathcal{D}_{t+1}, C_{t+1}$, or $\mathcal{H}_{t+1}$.

This formulation separates three questions that are often mixed together in the literature. Evaluation asks what has been observed. Orchestration asks how the observation should be used. Execution asks what object is actually changed. A method belongs to the core scope of this survey when this signal-to-decision-to-update path is explicit enough to show how evaluation changes a later data-related state. Measurement-only studies are treated as background evidence unless their outputs are used to change a downstream data-related object or control decision. Figure 3 also marks the distribution that the loop must not consume: an audit distribution held outside the update operator, which Section 4 develops as the separation between a controller-facing selection distribution and an untouched audit distribution.

Evaluation, orchestration, and execution are functional roles rather than mutually exclusive architectural modules. The same component may play multiple roles in a concrete system. For example, a learned judge performs evaluation when it scores candidate outputs, but it performs orchestration when its verdict determines whether candidates are retained, regenerated, or escalated. A teacher performs execution when it produces supervision, but it performs orchestration when student failures determine which examples or modalities receive teacher feedback. A memory system performs execution when it writes or deletes records, while its controller performs orchestration when task outcomes revise retrieval, replay, or promotion policies.

2.3. Literature Coverage and Survey Organization

We included works that either use evaluation feedback to update a data-related object, introduce reusable data/evaluation/control operators for recursive improvement, or analyze closed-loop risks such as contamination, reward hacking, evaluator dependence, model collapse, unstable updates, or irreproducibility. The resulting bibliography contains 300 unique cited works from 2020–2026, including 132 preprints or technical reports and 168 venue-resolved or archival publications. The most represented years are 2024–2026, and the largest venue groups are arXiv, ICLR, NeurIPS, ICML, CVPR, ACL, EMNLP, ECCV, AAAI, and TMLR.

The remainder of the survey follows the first three stages in Figure 4 as a signal–decision–update sequence and uses the fourth stage as a cross-cutting reliability audit. Section 3 studies evaluation signals and asks when measurement becomes actionable feedback. Section 4 studies orchestration and asks how signals are trusted, attributed, routed, scheduled, escalated, or stopped. Section 5 studies execution and asks how selected interventions update mutable data-related objects across pre-training, SFT, RL and preference optimization, OPD, and context or memory adaptation. Section 6 then analyzes why a higher score may still fail to establish reliable recursive improvement, focusing on signal validity, signal independence, update stability, data-support integrity, and system identifiability. Section 7 presents future directions worth exploring.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

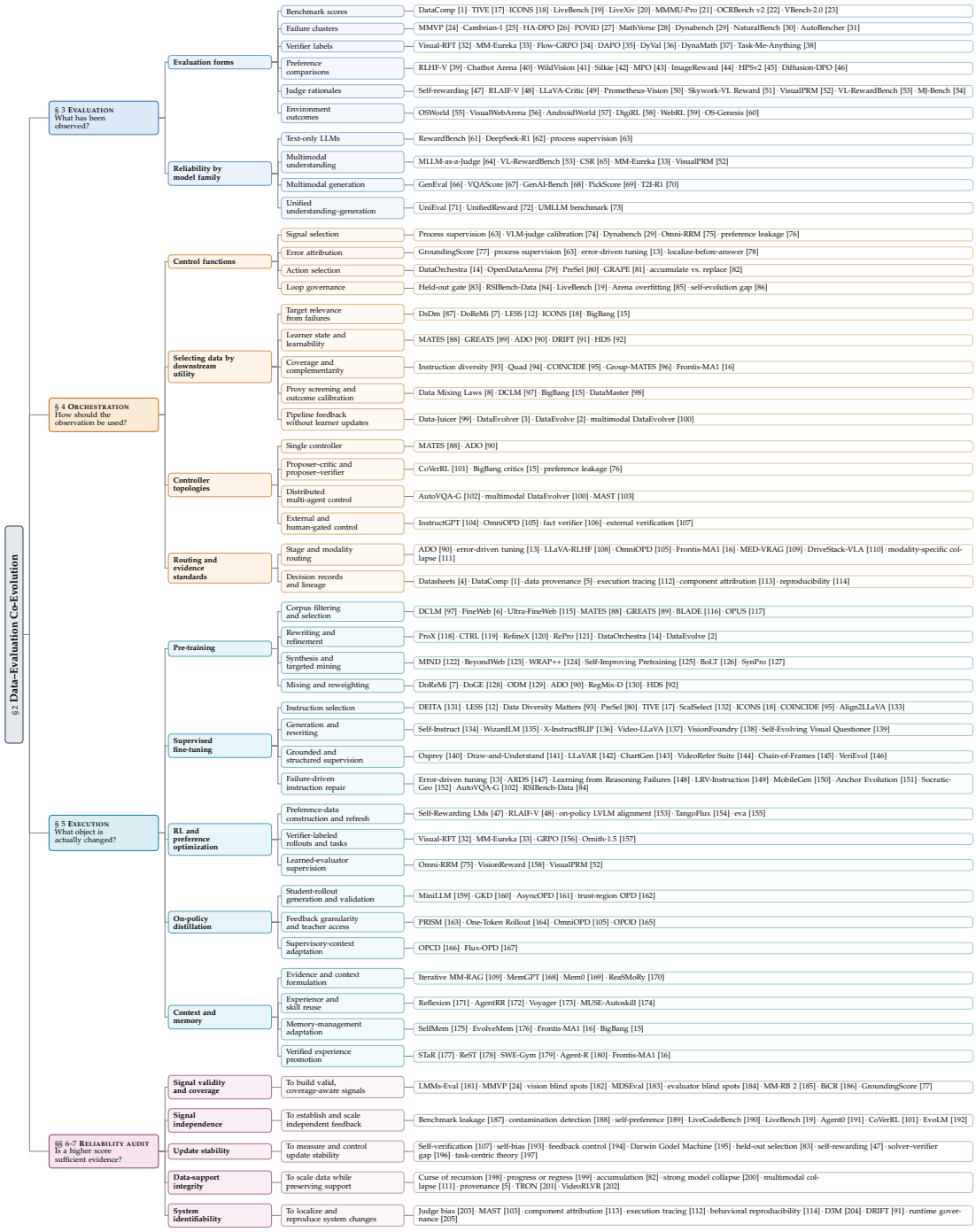


Figure 4. Taxonomy and reading map for data–evaluation co-evolution.

3. Evaluation Signals: From Measurement to Actionable Feedback

This section studies when an evaluation becomes an actionable signal for recursive improvement. An evaluation output contributes to the data–evaluation loop only when it does more than report performance: it must diagnose a model or system state and guide a downstream update to data, context, memory, training policy, or orchestration policy.

3.1. Evaluation Forms as Feedback Signals

We classify feedback by the form of the evaluation output rather than by the training stage that consumes it. The same signal can operate at multiple stages: a detector score can filter pre-training data, select SFT samples, or provide rewards for RL. Following this output-form view, we organize the literature into six recurring forms, each characterized by the mutable object it can change, the trust

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

assumption it introduces, and the failure mode it creates when wrong. Table 1 summarizes the six forms.

Table 1. Six evaluation forms as feedback signals in data–evaluation co-evolution. Each form is characterized by the loop signal it expresses, the mutable object it can change, the trust assumption it introduces, and the failure mode it creates when wrong.

Evaluation form	Loop signal	Mutable object updated	Trust assumption	Failure mode	Representative works
Benchmark score	Capability gap; mixture objective	Data mixture, curriculum, checkpoint selection	Transfer beyond the source benchmark	Benchmark overfitting	DataComp [1], ICONS [18], LiveBench [19]
Failure cluster	Weakness specification	Hard examples, synthetic data, negative samples	Correct causal attribution	Fixing the wrong bottleneck	MMVP→Cambrian-1 [24,25], POVID [27], NaturalBench [30]
Verifier label	Accept/reject; scalar reward	Rollouts, generated samples, reward-labeled data	Coverage and calibration	Reward hacking	Visual-RFT [32], Flow-GRPO [34]
Preference comparison	Pairwise winner–loser supervision	Preference pairs, reward-model data	Rater reliability and calibration	Bias propagation	RLHF-V [39], Chatbot Arena [40], ImageReward [44]
Judge rationale	Error explanation; revision target	Rewritten answers, critiques, process traces	Grounded validity	Fluent but false supervision	RLAIF-V [48], LLaVA-Critic [49], VisualPRM [52]
Environment outcome	Task success; tool feedback	Agent traces, memory, curriculum tasks	Environment transfer	Environment overfitting	OSWorld [55], DigiRL [58], OS-Genesis [60]

3.1.1. Benchmark Scores

An aggregate benchmark score is the coarsest and cheapest evaluation signal: almost every benchmark emits one. When consumed as a capability gap or a mixture objective, it can change dataset-level mutable objects, such as domain or modality mixtures, curricula, and checkpoint-selection policies. DataComp[1] provides a clean example. With the model, code, and compute fixed, a suite of 38 downstream tasks ranks candidate corpora, so the benchmark score selects datasets rather than models. Benchmark-derived scores can also guide sample-level selection. TIVE [17] and ICONS [18] estimate the influence of instruction examples on validation benchmarks and retain the highest-valued subset; the same signal is consumed at the SFT stage (Section 5.2) and, at corpus scale, in pre-training mixture search (Section 5.1).

The trust requirement is transfer: gains induced by the score must extend beyond the benchmark that supplied it. When they do not, the loop overfits the benchmark signal, often without directly training on any test item. This makes the failure harder to detect than ordinary contamination: the data update may favor examples, formats, or skills that are close to the benchmark without improving broader capability.

Benchmarks have therefore started to evolve in order to preserve the usefulness of their scores as feedback signals. LiveBench [19] refreshes its questions monthly and scores only objectively checkable answers. LiveXiv [20] generates VQA examples from newly posted arXiv figures and estimates model scores from a small subset, making frequent refresh affordable. When benchmark suites become saturated, successor versions such as MMMU-Pro [21], OCRBench v2 [22], and VBench-2.0 [23] restore headroom.

3.1.2. Failure Clusters

A failure cluster is a group of evaluation cases on which a model repeatedly fails for a shared reason, such as hallucinated objects [26,27] and shortcut use [28,206]. Unlike an aggregate score, a failure cluster identifies a weakness that may guide a data intervention. It suggests what evidence the next round should add, reweight, or contrast. MMVP [24] exposed clusters of CLIP-blind image pairs. Cambrian-1 [25] responded with vision-centric data and model-design changes, later evaluated with CV-Bench. Hallucination taxonomies play a similar role for alignment data: HA-DPO [26] and

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

POVID [27] synthesize dispreferred responses that instantiate measured hallucination patterns, turning benchmark-defined error categories into preference-optimization data.

The trust requirement is causal attribution: the cluster must identify the bottleneck that a data update can plausibly repair. A cluster that mixes perception failures with language-prior overrides may send data synthesis toward the wrong target. Ablation protocols such as MathVerse's [28] information variants and MMStar's [206] vision-indispensability filter therefore matter because they test whether the failure actually depends on missing visual evidence, reasoning, or shortcut use before it is converted into training data [28,206].

Failure clusters also show why benchmark construction and data construction are tightly coupled in recursive improvement. Dynabench [29] pays annotators for model-breaking examples, Natural-Bench [30] keeps natural images that models fail but humans pass—a failure-mining algorithm whose output happens to be a test set. AutoBencher [31] searches for difficult and novel question sets. The same mined failures can either remain held out as evaluation evidence or be admitted into the next training round as repair data. The boundary is therefore not the mining procedure itself, but the governance decision over whether a failure cluster is used for testing, training, or both.

3.1.3. Verifier Labels

A verifier label is a pass/fail decision or scalar score produced by a rule, program, tool, or fixed evaluator, such as exact answer match, IoU against a reference box, an object detector's verdict, or execution success. Among evaluation forms, verifier labels are often the most auditable because they can be recomputed and inspected. When consumed by a loop, they directly change sample-level mutable objects: which rollouts receive reward, which generated samples are retained, which prompts are filtered, or which web pairs enter a corpus.

At pre-training scale, thresholded CLIP scores and trained filtering networks decide corpus membership [1,207,208] (Section 5.1). At the RL stage (Section 5.3), verifiable rewards moved quickly from text [62] to multimodal models: MM-Eureka rewards answer matches [33]. Visual-RFT [32] lifts IoU and classification accuracy directly into GRPO rewards. Flow-GRPO [34] rewards a text-to-image model with the GenEval evaluator itself, raising SD3.5-M from 63% to 95% on GenEval. The same labels double as difficulty estimates: prompts whose rollouts all pass or all fail carry no gradient, so DAPO [35] filters them online.

The trust requirement is coverage and calibration: the verifier must check the property that the loop intends to improve, and its score must remain meaningful under optimization. When this assumption fails, the loop can reward-hack the verifier. Training directly against GenEval, for example, makes GenEval less independent as a measure of generalization: a higher GenEval score may partly reflect fit to the verifier rather than broader improvement, which is why held-out suites and reward-hacking analyses are needed alongside optimized verifier scores [70].

This risk also explains why benchmark evolution around verifier labels often takes the form of generated or regenerated tasks. DyVal [36] composes reasoning tasks from graph structures. DynaMath [37] renders each seed problem as a program that emits fresh visual variants and reports worst-case accuracy. Task-Me-Anything [38] emits large numbers of verifiable task instances on demand. Regenerated variants can restore score gaps erased by memorization [209]. Because these generators produce their own ground truth, they can support both evaluation and training; the governance question is how to keep the instances used for optimization separate from those used for audit.

3.1.4. Preference Comparisons

A preference comparison states, at minimum, that one output is preferred to another under a given prompt, context, or criterion. When consumed by a loop, this signal changes preference pairs, reward-model training data, or winner–loser examples used by preference-optimization methods (Section 5.3).

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Preference sources form a cost–fidelity spectrum. At the high-fidelity end, RLHF-V [39] collects segment-level human corrections for hallucinated multimodal answers and converts them into dense preferences. At scale, arena votes are produced as a by-product of evaluation: Chatbot Arena [40] and WildVision [41] rank models live while accumulating preference corpora for later training. Between these endpoints, preferences are manufactured or semi-automated: Silkie [42] uses GPT-4V to annotate a feedback corpus, and MPO [43] builds pairs from answer matching and image-ablated continuations. For multimodal generation, reward models such as ImageReward [44], PickScore [69], and HPSv2 [45] distill human preferences into signals that both rank generators and support reward fine-tuning or Diffusion-DPO [46].

Preference comparisons make the evaluation–training overlap explicit: the same vote, pair, or reward label can rank systems and become mutable supervision for the next update. The trust requirement is rater reliability and calibration. The failure mode is bias propagation: position bias, verbosity bias, style preference, model-family preference, or weak visual grounding can be copied into every pair the rater labels and then amplified by preference optimization. Arena votes [40,41] make this overlap especially visible. They are evaluation signals because they rank systems on fresh user prompts, but they also become training signals when the same votes are accumulated into preference corpora. This dynamic stream tracks the moving frontier better than a frozen preference test set, but using it for training requires attention to prompt distribution, rater population, data access, and independence from final evaluation.

3.1.5. Judge Rationales

A judge is an evaluator, usually a learned model, that assesses candidate outputs and returns a score, preference, accept/reject decision, or critique. Here we focus on cases where the judge returns a rationale rather than only a verdict: it may explain what is wrong, identify the violated criterion, localize an unsupported step, or propose a revised answer. This makes judge rationales a scalable but difficult-to-verify feedback form for open-ended outputs. When consumed by a loop, they change mutable supervision objects such as rewritten answers, critique corpora, process traces, preference annotations, data-selection scores, or process-reward labels. Examples include judge-annotated corpora for preference training (Section 5.3), critic models whose scores select data or policies [49–51], and process reward models that convert step-level judgments into supervision [52].

The limiting case is self-referential feedback, where a model judges its own outputs and trains on the result [47]. This design reduces annotation cost but couples the generator and evaluator. In multimodal settings, the coupling is more dangerous because the judge may inherit the same perceptual blind spots or hallucinations it is supposed to detect. CSR [65] mitigates this risk by re-anchoring self-reward in image-conditioned likelihood, while RLAIF-V [48] replaces direct self-judgment with decomposed peer verification.

The trust requirement is grounded validity: the rationale must be supported by the evidence it cites, whether that evidence is text, image regions, video frames, audio, documents, tools, or environment state. The failure mode is fluent but false supervision. An MLLM judge may confabulate image content and then score or revise an answer against that confabulation [64]. Once such rationales enter training, they can corrupt preference pairs, revised answers, filters, and process traces downstream.

Because judges and reward models are themselves trained artifacts, this signal requires second-order evaluation: the field must evaluate the evaluators before their outputs are used as feedback. RewardBench established this practice for text [61], and multimodal extensions such as VL-RewardBench, MJ-Bench, Multimodal RewardBench, VideoRewardBench, and CMI-RewardBench audit reward and judge behavior across vision-language, generation, video, and audio settings [53,54,210–212]. These benchmarks matter for co-evolution because an uncalibrated judge can silently corrupt every preference pair, reward, filter, or rationale that depends on it.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

3.1.6. Environment Outcomes

An environment outcome is the success or failure of a whole interaction, such as completing a computer-use task, placing a web order, executing a tool call, or reaching a verified task state. OSWorld [55], VisualWebArena [56], and AndroidWorld [57] define such outcomes with executable checks in real or realistic environments. When consumed by a recursive loop, the outcome changes trajectory-level mutable objects: agent traces, replay buffers, curriculum tasks, memory entries, tool-use records, or instruction trajectories. DigiRL [58] trains device-control agents with RL against an autonomous environment evaluator. WebRL [59] turns failed trajectories into new curriculum tasks for web agents. OS-Genesis [60] explores the environment, synthesizes instruction trajectories, and filters them with a trajectory reward model. Runtime adaptation consumes the same signal when task success determines which experiences are written to memory, retained, or promoted into later training data (Section 5.5).

The trust requirement is environment transfer: success in the training or evaluation environment must predict success on fresh task instances, unseen layouts, new websites, or real deployment settings. The failure mode is environment overfitting. An agent may memorize a site layout, exploit a simulator artifact, or learn shortcuts that satisfy the environment checker without solving the intended task. This risk changes what benchmark evolution means for environment outcomes. Because an agent can overfit fixed trajectories, layouts, or simulator artifacts, environment benchmarks often need to evolve by regenerating task instances rather than only refreshing question sets. AndroidWorld parameterizes tasks so each run can be instantiated freshly [57], reducing direct memorization of fixed trajectories. In such loops, evaluation infrastructure can also become training infrastructure: the same environment may produce rewards, generate traces, define curricula, and audit success. Environment outcomes therefore make the governance problem explicit: systems must separate the instances used for training or adaptation from those reserved for held-out evaluation.

3.2. Signal Reliability Across Model Types

The six feedback forms above are available across model families, but their reliability changes with the model being improved. The central pattern is a reliability gradient: *the more multimodal, generative, and unified a system becomes, the harder it is for evaluation outputs to serve as trustworthy training signals*. A recursive improvement loop that works for one model family therefore cannot usually be transferred to another by reusing the same signal source; the signal must be revalidated against the evidence, output space, and failure modes of the target model.

Text-only LLMs.

Text-only LLMs provide the most favorable case. Many outputs can be checked by rules or programs, culminating in RL with verifiable rewards [62]. Less verifiable behavior can be routed through reward models, audited by RewardBench [61], or refined with process reward models [63]. The main problems are signal integrity—contamination, reward overoptimization, and judge bias—rather than signal availability.

Multimodal understanding models.

Multimodal understanding models inherit the text-side stack but add an evidence-use requirement. A signal must verify not only whether the answer is correct, but also whether it depends on the relevant image, region, OCR token, chart structure, video frame, or document evidence. This narrows the verifiable slice and weakens learned judges, which may share the policy's perceptual blind spots [53,64]. Current pipelines therefore combine partial verifiers, claim decomposition, visual-likelihood calibration, and multimodal process reward models [32,33,48,52,65].

Multimodal generation models.

Multimodal generation removes the assumption of a single correct output. Signals therefore become either *reductions*, which project an output onto checkable properties such as object presence

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

or VQA-based alignment [66–68], or *preferences*, which provide broader but more overoptimizable judgments [44,45,69]. Reductions are auditable but gameable; preferences are holistic but easier to exploit. Generation RL pipelines therefore need multiple rewards and reward-hacking analysis [34,70].

Unified understanding–generation models.

Unified models add a commensurability problem. An MMMU accuracy, a GenEval pass rate, and a preference margin do not share a natural scale, yet loops need comparable signals for checkpoint selection, data-mixture balancing, reward aggregation, or stage scheduling. Current responses probe understanding–generation consistency directly, use one branch to evaluate the other, or train reward models across both task families [71–73]. The remaining risk is that self-judgment, metric incompatibility, or cross-task interference can make an apparent gain hard to attribute. Section 6 analyzes the resulting loop-level problems through signal validity, signal independence, update stability, data-support integrity, and system identifiability.

4. Orchestration Decisions: From Feedback Signals to Data Interventions

Recent surveys distinguish the feedback observed by a self-improving system from the component or state that is subsequently adapted [11]. Multimodal self-improvement further spans training data, preference feedback, and inference-time adaptation rather than one uniform update path [213]. Section 3 defines the behavioral evidence available to the loop, but a failure signal does not determine its own intervention: the same failure cluster can motivate collecting new pre-training documents, constructing a supervised fine-tuning (SFT) repair set, refreshing preference pairs, requesting teacher feedback, or changing runtime memory. Orchestration is the policy that resolves this underdetermination.

Existing self-evolution taxonomies commonly organize systems by the role or component being changed [11]. We instead treat orchestration as a *data-decision problem*, rather than as an inventory of teachers, judges, critics, or agents. At minimum, a stage-local rule performs orchestration when a feedback verdict changes which data are retained, rejected, or routed; adaptive orchestration is the stronger case in which accumulated feedback changes the target, operator, allocation, gate, or future decision policy. This section focuses on that adaptive case. We specialize the general orchestration action in Eq. 2 to an explicitly data-centric decision:

$$a_t = (z_t, o_t, \pi_t, b_t, g_t), \tag{4}$$

where z_t identifies the mutable data object and execution stage, o_t chooses a data operator and source, π_t allocates mass across examples, capabilities, domains, or difficulty levels, b_t assigns collection, training, and evaluation budget, and g_t specifies acceptance, escalation, stopping, or rollback. DataMaster exposes source, composition, and transformation as alternative branches of a mutable data state [98]. DataOrchestra exposes a complementary per-example choice among dropping, retaining, and cleaning a chunk [14]. The tuple is our survey abstraction of these decisions: given the current learner and its downstream failures, which feasible change to the data state has the highest credible marginal value?

We first develop signal selection, error attribution, action selection, and loop governance as decisions over data. We then examine downstream-utility estimation, compare representative systems by the point at which feedback closes their loop, and discuss controller topologies. The final parts route decisions across training stages and modalities and specify the evidence needed to audit an orchestration claim.

4.1. A Data-Centric Decision Space

BigBang separates critic evidence, frontier-task demand, synthesis strategies, and promotion by downstream outcomes even though they participate in one loop [15]. DataMaster similarly separates its DataTree controller from the Data Pool and Global Memory that record candidate and realized data states [98]. We abstract these designs into four coupled functions: signal selection determines

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

which observations may enter the decision; error attribution turns those observations into a data demand; action selection chooses a target, operator, source, and allocation; and loop governance decides whether the proposed transition should be committed. They may be implemented by one controller or distributed across several components. Each function should leave an observable trace in the affected data, control state, loop history, or decision log, including when the result is abstention or no update.

4.1.1. Signal Selection: Building a Decision-Grade Evidence Set

Process supervision demonstrates that feedback may localize an error at a reasoning step rather than only at the final answer [63]. Vision–language model (VLM) judge studies show that the same evaluator can support ranking while remaining unreliable for absolute thresholding [74]. These examples define a heterogeneous decision-evidence space: aggregate scores can set a capability-level acquisition budget, verifier labels can gate individual rollouts, judge rationales can suggest a rewrite, and environment outcomes can decide whether a trajectory is retained. Because these signals differ in granularity, coverage, cost, and bias, a controller should preserve their provenance and uncertainty rather than collapse all evidence into one scalar reward.

Dynamic benchmarking prefigured this shift from measurement to intervention. Dynabench placed humans and the current model in a repeated data-collection cycle, so observed failures changed which evaluation examples were acquired next [29]. Once an evaluator is allowed to control a data gate, however, its validity becomes action-dependent: scaling-law analyses of reward-model overoptimization show that increasing a learned proxy can eventually reduce the gold objective [214]. The training data used to construct the gate are themselves mutable. Omni-RRM, for example, constructs rubric-grounded preference pairs and dimension-wise justifications for image, video, and audio reward-model training, with text evaluated as a transfer setting [75]. This automates evaluator-data construction, but it does not by itself establish that the resulting gate remains calibrated after repeated use.

Signal form and decision role must be considered jointly. VLM-judge uncertainty varies by task and use, so an evaluator that supports model comparison may remain unsafe as a per-example data gate [74]. Signal selection is therefore curation of decision evidence: a narrow verifier should abstain outside its support, and disagreement should remain as metadata so that unsupported examples can be routed to another verifier or human review.

The main risk is that a cheap, correlated, or contaminated signal becomes the de facto objective of the data loop. Preference leakage demonstrates a concrete dependence channel: judges favor outputs from generators that are the same model, an ancestor, or a member of the same model family [76]. Reusing such a signal can progressively concentrate the dataset around its blind spots. Signal provenance, abstention, and an evaluation channel not exposed to the controller must therefore be enforced within the orchestration policy rather than treated as evaluation hygiene outside the loop.

4.1.2. Error Attribution: From Failure to Data Demand

Error attribution maps decision-grade evidence to a demand profile over capabilities, domains, modalities, difficulty levels, and training stages. A useful diagnosis is not merely "the score is low," but, for example, "the current model fails medium-difficulty document questions because relevant regions are not grounded before reasoning." Such a diagnosis can increase demand for region–text alignment examples without indiscriminately adding more reasoning traces. Process supervision illustrates how localization at the reasoning-step level can make a different supervision object actionable [63].

Attribution must distinguish the location where a failure appears from the data mechanism that caused it. An optical character recognition (OCR) error may reflect missing pre-training coverage, poor SFT instruction grounding, a retrieval failure, or an evaluator that misread the image. GroundingScore addresses the last ambiguity by explicitly checking the visual premises used by a process reward model before treating its step score as evidence of reasoning quality [77]. Localizing Before Answering makes the relevant image region an explicit evaluation object, supplying finer evidence than an answer-

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

level hallucination label [78]. A self-evolving visual concept library goes one step further by using vision–language critics to revise the concept substrate that later predictions consult [215].

Controllers can reduce causal ambiguity by comparing failure slices, intermediate traces, modality ablations, and prior interventions recorded in the loop history. Error-driven multimodal tuning operationalizes this mapping: a teacher identifies erroneous reasoning steps, summarizes missing skills, and uses those skills to retrieve a targeted repair subset [13]. The attribution output should therefore be a ranked set of repair hypotheses, each paired with the data object it predicts would change the outcome.

The same mapping can drive synthesis rather than retrieval. Learning from Reasoning Failures uses a stronger model to analyze a weaker multimodal learner's errors, propose examples targeted to the inferred failure modes, and filter the resulting repair set [148]. It demonstrates an attribution-to-data path, but not an outcome-controlled loop: the reported post-training evaluation does not return to revise a later round of failure analysis.

Error-driven multimodal tuning shows that the diagnosed skill determines which repair examples are retrieved, so a mistaken diagnosis can redirect an otherwise competent data generator toward the wrong bottleneck [13]. GroundingScore likewise shows that an apparent reasoning error can originate in an unsupported visual premise [77]. Because several interventions can yield the same aggregate score change, causal attribution should remain provisional until a controlled data change validates it [1]. This training-data attribution question—which data intervention repairs behavior—is distinct from system identifiability, which asks which running component caused an observed change [113]. An auditable controller should record the two as separate hypotheses rather than silently converting a component diagnosis into a data prescription.

4.1.3. Action Selection: Choosing the Data Object, Operator, and Allocation

DataOrchestra demonstrates that one action space can include dropping, retaining, and conditionally cleaning individual chunks [14]. OpenDataArena demonstrates decisions over difficulty composition and multi-domain mixtures at dataset scale [79]. We use these concrete designs to instantiate z_t, o_t, π_t, and b_t in Eq. 4. The target z_t can be a corpus subset, instruction–response pair, preference comparison, rollout trace, evaluator-training example, retrieval record, or promotion queue. The operator o_t selects, rejects, rewrites, synthesizes, relabels, deduplicates, reweights, reorders, stores, or deletes; π_t controls relative frequency and sequencing; and b_t chooses between further evidence collection, candidate generation, a training probe, and a full update.

PreSel provides a cost-aware ordering example: it selects unlabeled images and allocates task-specific budgets before instruction generation, avoiding annotation and tuning over the full image pool [80]. This is a pre-generation primitive rather than an adaptive loop, but together with DataOrchestra's select-or-repair action space it motivates comparing selection, repair, and synthesis instead of defaulting to generation. Filtering, ranking, mixture adjustment, repair, and pre-generation selection are action primitives; they become an adaptive orchestration loop only when evidence from their outcomes changes a later target, allocation, gate, or policy.

GRAPE shows that optimizing a mixture for a single target can improve that objective while degrading other targets [81]. The decision should therefore record both its intended target and protected slices on which regression is unacceptable. Recursive synthetic-data experiments provide a complementary data-state warning: replacing the original real data at each round caused collapse in the studied settings, whereas accumulating generated data alongside the original pool avoided that outcome [82]. Without such constraints, a controller may oversample conspicuous hard cases or amplify mislabeled examples; modality-specific collapse further shows that slower or weaker channels need explicit protection [111]. Support preservation and update stability must therefore be constraints on action selection itself.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

4.1.4. Loop Governance: Acceptance, Stopping, and Rollback

Recursive Self-Evolving Agents commits a changed context artifact only when it does not regress on a disjoint held-out split [83]. RSIBench-Data shows why the associated history matters: 18 of 23 searches that continued beyond an observed peak ended with a lower-scoring final attempt [84]. We abstract these cases as g_t, which can gate an individual sample, batch, data pipeline, or checkpoint. A complete policy records the trajectory, escalation and stopping decisions, and a best-so-far state so that promotion and rollback occur at the same granularity as the update.

LiveBench motivates refreshing benchmark instances when model developers may have observed earlier test content [19]. Analysis of Chatbot Arena further identifies overfitting incentives from repeated private testing, selective disclosure, and asymmetric access to platform data [85]. Adaptive selection creates a related dependence: repeatedly using a downstream slice to choose data turns that slice into controller information even without gradient overlap. Preference leakage shows that dependence can also arise through model identity and lineage [76], while strictly self-generated reasoning can retain a gap to oracle supervision [86]. Governance should therefore separate controller-facing selection from untouched audit data; independent evidence and rollback are complementary rather than interchangeable safeguards.

4.2. Selecting Data by Downstream Utility

DsDm defines data quality through the target performance of the model produced by a specified learning algorithm rather than through an intrinsic property of records [87]. This motivates our central co-evolution problem: choose among candidate data interventions using evidence from the current learner. Let $\mathcal{A}_t$ denote feasible actions, M_t the current model or model-based system, and M_t^a the system obtained after executing action a. We use the following idealized downstream-utility abstraction for a controller-facing selection distribution $\mathcal{Q}_t^{\mathrm{sel}}$:

$$\Delta_{\mathrm{down}}(a) = J(M_t^a; \mathcal{Q}_t^{\mathrm{sel}}) - J(M_t; \mathcal{Q}_t^{\mathrm{sel}}), \tag{5}$$

OpenDataArena reports several value and dataset-analysis dimensions rather than reducing data utility to one quality label [79]. Accordingly, J is a pre-specified scalarized, possibly risk-adjusted downstream utility, while the full capability, safety, calibration, and efficiency vector is retained for constraint and Pareto analysis. A cost-aware controller can then use

$$a_t^* \in \arg\max_{a \in \mathcal{A}_t} \left[\widehat{\Delta}_{\mathrm{down}}(a) - \lambda_t \, \mathrm{Cost}(a) \right], \tag{6}$$

where λ_t controls the trade-off between estimated gain and cost. The optimization is subject to coverage, provenance, safety, and protected-slice regression constraints. The Data Provenance Initiative demonstrates why source, licensing, and downstream-use records must remain attached to selected data [5]. DataComp demonstrates the complementary need to hold training and compute fixed when comparing candidate datasets through downstream outcomes [1]. Here $\mathcal{Q}_t^{\mathrm{sel}}$ is the controller-exposed evaluation distribution; an adaptive evaluation queue is recorded in the control state or loop history, while a fixed $\mathcal{Q}^{\mathrm{audit}}$ is never used for data selection, controller calibration, or policy updates. These equations are our survey abstraction of how systems estimate counterfactual value without fully training every candidate.

The literature instantiates this counterfactual at several fidelities: LESS uses optimizer-aware influence [12], Data Mixing Laws uses proxy training runs [8], and BigBang returns to full post-update evaluation [15].

An optimal-control view makes the sequential dependence explicit: a useful example changes the learner state against which later examples are valued. PDS formulates pre-training selection as a generalized optimal-control problem and derives conditions linking the selection decision to language-model training dynamics [216]. It remains an offline selector rather than a recursive feedback policy, but it clarifies why a one-time, pointwise quality score is an incomplete approximation to Eq. 6.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

4.2.1. Target Relevance from Downstream Failures

Selection first needs a target distribution. A deployment workload, held-out task set, or capability profile can convert failures into weights over skills and domains; candidates are then valued by how well they address that demand. DoReMi is an important domain-level contrast: it uses group distributionally robust optimization (Group DRO) on a proxy model to learn domain weights without downstream-task knowledge, so it provides a proxy-optimized mixture-allocation baseline rather than failure-conditioned target relevance [7]. LESS estimates the alignment between candidate instruction data and a downstream target through optimizer-aware influence, providing a representative one-pass approximation to target relevance [12]. ICONS extends target-aware influence selection to vision–language mixtures and aggregates task-level influence through consensus across validation tasks [18]. BigBang uses performance on held-out real research tasks to identify capability gaps and adjust the domain distribution, difficulty, and capability requirements of subsequent synthetic tasks, closing target-relevance estimation into an iterative loop [15].

Across these cases, target relevance is narrower than general data quality: an example for an already-saturated skill may have less marginal value than a verifiable example covering a current failure. GRAPE further shows that single-target mixture optimization can improve one objective while degrading other targets [81]. The demand profile should therefore include protected coverage and distinguish $\mathcal{Q}_t^{\text{sel}}$, which the controller may adapt to, from an untouched $\mathcal{Q}^{\text{audit}}$.

4.2.2. Learner State, Difficulty, and Marginal Learnability

The value of an example changes as the learner changes. Static quality filtering ignores whether the current model already masters the example, cannot yet learn from it, or would benefit from a nearby prerequisite. At example granularity, MATES uses local probes from the current model to learn a data-influence estimator and refresh selection as pre-training proceeds [88]. GREATS moves the refresh inside training by selecting data at every optimization iteration [89]. At domain granularity, ADO estimates each domain's remaining learning potential with online scaling laws and adjusts the mixture concurrently with training, without a separate proxy model [90]. DRIFT supplies the analogous instruction-data view by using current-model rollouts as validation anchors for attribution [91]. HDS then illustrates that the scheduler may need to negotiate quality, difficulty, diversity, and domain objectives rather than impose one universal ranking [92]. Despite their different granularities and update schedules, their common contribution is to make utility local to the current learner state.

MobileGen operationalizes a capability frontier by separating structural and semantic difficulty before generating new GUI trajectories [150]. Task-centric theory gives conditions under which moderate difficulty and easy-to-hard scheduling improve finite-sample guarantees over fixed mixtures [197]. Difficulty should therefore not be equated with utility: easy data can be redundant, while very hard data can be unverifiable or outside the learner's present support. MATES also makes clear that probe-based utility is conditional on a current model and optimizer state [88]; it should be refreshed after material updates and treated as local evidence rather than a permanent ranking.

Curriculum systems move one step further by changing the candidate distribution itself. WebRL generates new web-agent tasks from unsuccessful attempts, making current failures a source for the next online curriculum [59]. MobileGen profiles structural and semantic capability frontiers, then uses the profile to update a challenge-centered difficulty distribution for trajectory synthesis [150]. Ouroboros-Spatial operates at a finer granularity: it uses the solver's per-sample confidence to steer a frozen proposer toward spatial questions matched to the solver's evolving ability [217]. A task-centric analysis supplies a complementary theoretical view: under explicit conditions on initialization, difficulty, and sample budget, easy-to-hard self-improvement can dominate a fixed mixture, while the feedback loop still predicts eventual saturation [197]. Together these works distinguish a fixed curriculum from learner-conditioned data demand; because self-generated supervision can preserve a gap to oracle feedback, learner-derived success signals can inherit the learner's blind spots [86].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

4.2.3. Coverage, Complementarity, and Exploration

Data Diversity Matters treats instruction diversity as a robustness objective rather than an incidental by-product of quality filtering [93]. COINCIDE measures vision–language coverage through concept–skill transferability rather than raw visual dissimilarity [95]. At selection time, Quad combines influence, clustering, and exploration to reduce the redundancy of taking only the highest-scoring records [94]. Together these works motivate complementarity across capabilities, domains, reasoning patterns, modalities, and trajectory states.

Group-MATES makes the unit of utility a selected set rather than an isolated record: it collects reference-loss influences along sampled training-data trajectories and scores each candidate conditional on the previously selected set [96]. This provides a bridge between pointwise influence and set-level coverage, although the learned relationships are local estimates rather than identified causal complementarities.

Frontis-MA1 makes this trade-off explicit in executable machine-learning engineering. At inference, it selects candidate program nodes using quality, parent-relative progress, and method-family novelty, then retrieves bounded, operator-conditioned evidence from their experience cards. The Draft, Improve, Debug, and Crossover interfaces trained with execution-grounded SFT and reinforcement learning (RL) are composed again during long-horizon search [16]. Its experience cards are therefore mutable runtime data for orchestration; the remaining limitation is that fixed hand-designed search weights leave the relative value of quality, progress, and novelty task-independent.

RSIBench-Data shows the opposing risk to this exploration policy: continuing a search after its observed peak frequently yields a worse final attempt [84]. Exploration is therefore not automatically beneficial. A credible policy should report its exploration budget, the coverage measure that prevents collapse, and the evidence that terminates an unproductive branch.

4.2.4. Proxy Screening and Outcome Calibration

Data Mixing Laws fits small-scale mixture runs as a lower-cost proxy for unseen target-scale mixtures [8]. DCLM instead fixes the training recipe and downstream evaluation to compare curation procedures at scale [97]. These are distinct evidence levels: the former screens candidate allocations, whereas the latter isolates data-pipeline differences under controlled training. Outcome-based recalibration requires the feedback path illustrated next.

BigBang is a clear instance of this outcome-calibrated pattern. It uses critics as fast proxies and periodically selects representative synthesis pipelines for higher-fidelity validation. Model variants trained on their outputs are evaluated on held-out real tasks, and a meta-critic uses discrepancies between critic assessments and observed training outcomes to revise evaluation criteria and generation strategies [15]. DataMaster holds the learning algorithm fixed so that downstream results compare alternative data-engineering branches as a whole, although this comparison does not by itself isolate the contribution of an individual discovery, selection, cleaning, or transformation step [98].

DataComp provides a precedent for isolating the data contribution under fixed training and compute [1]. For adaptive selection, an analogous experiment would compare candidate policies under matched budgets and reserve an audit distribution outside the controller; without that design, evidence supports the selected pipeline as a whole rather than a causal claim about its utility estimator.

4.2.5. Pipeline-Level Feedback without Full Learner Updates

The progression from data tooling to orchestration is a progression in what feedback may change. Data-Juicer established a configurable operator and recipe abstraction, together with model-based evaluation of alternative recipes, but the user still specifies the processing policy [99]. Later systems make per-example plans, revise the preparation policy from trial outputs, or let post-training outcomes alter the next data strategy. Table 2 separates these feedback depths; sharing an operator library does not make their recursive boundaries equivalent.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Full retraining is not the only way to make data preparation adaptive. The automatic data-preparation DataEvolver builds a profile from sampled raw data, fixed high-quality seeds, and accumulated experience; expands an operator library; assembles operators into an executable directed acyclic graph (DAG); and tests candidate pipelines on trial subsets. Logical and execution checks establish whether a plan can run; discrepancies between trial outputs and the seed examples update the profile and experience memory used by later plans [3].

Within this design, logical, execution, and seed-discrepancy checks update the preparation plan, whereas downstream SFT is validation after the inner search. It is therefore a preparation-policy loop, not a fully closed learner–data loop. Seed-level feedback makes search cheaper, but learned-proxy overoptimization shows why proxy improvement need not guarantee downstream utility [214]. Periodic downstream checks are needed to test when that relationship fails.

DataEvolve closes a related loop at the curation-strategy level: judge scores and diagnostics on sampled original–cleaned document pairs revise per-category strategies before selected strategies are applied at corpus scale [2]. Its later pre-training results validate the searched strategy, whereas BigBang makes post-training outcomes part of the controller for subsequent synthesis [15]. Keeping these recursive boundaries explicit is more informative than labeling both systems simply "self-evolving."

4.3. Representative Adaptive Data-Orchestration Systems and Their Recursive Boundaries

The recursive boundary is the highest-level outcome that feeds back to change a later data decision. DataOrchestra represents input-conditioned, per-example planning without a reported online policy update [14]. BigBang represents the stronger case in which post-training outcomes control later synthesis and evaluation strategies [15]. Table 2 organizes representative systems between these boundaries by the decision that changes data and the point at which feedback returns. This prevents all automated data pipelines from being described as equivalent closed loops.

The table's contrast between preparation-policy recursion and downstream-controlled recursion does not imply an evidence ranking: a full loop can entangle changes in data, synthesis strategy, critic criteria, and training. Repeated adaptation can also expose the controller's selection benchmark to overfitting [85]; stronger causal evidence still depends on controlled comparisons [1]. Thus, "representative" means that a mechanism clarifies the design space, not that it has broad reproduction or field consensus.

The explicit state separation summarized in Table 2 suggests a modular research direction: future systems can expose and separately calibrate the target profile, utility estimator, exploration rule, and promotion gate in Eq. 4. Such factorization makes a pipeline easier to compare, ablate, and transfer while preserving its data-centric contribution.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Table 2. Representative data-centric orchestration systems. The table emphasizes the feedback-to-decision-to-data path and states where recursion closes; it is not an exhaustive catalogue of automated data construction.

Representative work	Feedback used	Data decision	Mutable object	Loop closure
DataEvolve [2]	LLM-judge scores and diagnostics on sampled original–cleaned document pairs	Revise the curation strategy across iterations	Per-category strategy, experience and strategy pools, and sampled cleaned outputs	Curation strategy updates from sample feedback; pre-training only validates it
DataEvolver for automatic data preparation [3]	Logical and execution checks, then discrepancies between trial outputs and fixed high-quality seeds	Evolve operators, a preparation DAG, and feedback memory	Operator library, DAG, data profile, experience memory, and trial outputs	Preparation policy updates from seed discrepancies; training only validates outputs
DataOrchestra [14]	Tool-execution feedback and stage-specific verifier judgments during offline plan evolution	Drop, leave untouched, or clean a chunk and select its processing stages	Chunk-level curation plan, rewriting instructions, and processed chunk	Offline feedback trains the orchestrator; no online policy update is reported
OpenDataArena-driven dataset engineering [79]	Value-anchored rankings and multi-dimensional analyses of candidate data	Construct difficulty-aware mathematics data and patch multi-domain SFT mixtures	Selected records, difficulty composition, and domain mixture	Rankings guide construction; downstream evaluation does not trigger another update
DataMaster [98]	Downstream training and evaluation outcomes under a fixed learning algorithm	Select the next data branch, source, composition, or transformation	DataTree, Data Pool, and Global Memory	Downstream outcomes update the DataTree; the learning algorithm stays fixed
BigBang [15]	Critic judgments and held-out real-task performance after training	Revise domain, capability, difficulty, synthesis, and evaluation policies	Frontier-task distribution, synthesis programs and strategies, and critic criteria	Post-training outcomes update critic and synthesis policies
Frontis-MA1 [16]	Execution status and scores; training-time reward, child variance, and visit cooling; search-time quality, progress, and novelty	Select parent programs and bounded operator-conditioned evidence, then apply a trained operator	Programs, trajectories, experience cards, and search board	Inference updates programs and experience, not model weights
Multimodal DataEvolver [100]	Fixed-verifier quality statistics and rejection causes summarized into round-level feedback	Revise retrieval queries and generation prompts, and synthesize for under-covered regions	Candidate and pass sets, query and prompt policies, targeted samples, and semantic-feedback memory	Verifier feedback updates construction policy; fine-tuning only evaluates data

4.4. Controller Topologies as Implementations

MAST shows that multi-agent failures arise from coordination and inter-agent misalignment as well as from individual reasoning errors [103]. This motivates treating teachers, judges, critics, planners, schedulers, tools, and humans as implementations of the preceding decision functions rather than as the taxonomy itself. Their topology matters because it determines which errors are shared, which decisions are inspectable, and where independent evidence can enter.

The literature also reflects a transfer of control over feedback data. Reinforcement learning from human feedback (RLHF) pipelines established human demonstrations and comparisons as external data for supervised and reward-guided policy updates [104]. Constitutional AI moved part of that path to model-generated critiques, revisions, and AI preferences constrained by a fixed constitution [218]. Self-Rewarding Language Models then used the improving model itself to produce rewards for later

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

iterative preference updates [47]. This progression raises scale but can weaken signal independence, which is why selective escalation and role separation matter more than the nominal number of agents.

4.4.1. Single-Controller Selection

MATES uses local probes from the current model to refresh example selection during pre-training [88]. ADO instead uses current per-domain loss trajectories to update mixture weights during training [90]. Both instantiate a single controller that consumes learner-state evidence and emits a new sampling decision. This topology is inexpensive and replayable, but a biased controller can recursively reshape the evidence on which it is recalibrated; learned-proxy overoptimization illustrates the resulting dependence risk [214]. Conservative update sizes, protected slices, and periodic recalibration are therefore central safeguards.

4.4.2. Proposer–Critic and Proposer–Verifier Separation

CoVerRL alternates generator and verifier roles to escape a consensus trap, but its use of one model for both roles also shows that role separation alone does not create an independent signal [101]. A proposer can synthesize or transform candidate data while a critic explains defects and a verifier decides whether evidence supports acceptance; the benefit comes from informational separation, not component count. An executable checker can reject a program using evidence unavailable to the generator, whereas a critic from the same model family may merely restate the proposal. BigBang's distinction between critic judgments and observed post-training outcomes illustrates how a higher-fidelity channel can recalibrate a cheaper one rather than simply vote with it [15].

This topology can still fail when the proposer and verifier share training data, reward shortcuts, or benchmark exposure. Preference leakage makes this correlation observable despite assigned role separation when the underlying models share identity, lineage, or family [76]. Repeated agreement is not evidence of independence. The controller should therefore record verifier coverage and abstentions and route unsupported cases to a tool, model family, or human gate with a genuinely different evidence source.

4.4.3. Distributed and Multi-Agent Control

AutoVQA-G separates fine-grained visual verification from a prompt-optimization agent that mines critiques of failed samples to revise subsequent grounded-VQA annotation [102]. This illustrates how a multi-agent system can distribute diagnosis, retrieval, generation, verification, and scheduling when the data unit itself is structured. The multimodal DataEvolver in Table 2 provides a second pattern: rejection causes and round-level failure statistics are distilled into semantic feedback that revises later retrieval and targeted synthesis rather than disappearing after filtering [100].

Automated multi-agent attribution shows that distributed trajectories make component-level causes harder to identify [113]. Agent messages may alter prompts without leaving a structured data diff, correlated agents can amplify one another's error, and distribution can impose nontrivial coordination cost. MAST's taxonomy confirms that failures arise from coordination and inter-agent misalignment as well as from an individual model's reasoning error [103]. An auditable controller should therefore expose role-specific observations, decisions, and writes to shared state. If those effects cannot be separated, the system should be reported as a composite pipeline rather than attributing gain to one orchestration mechanism.

4.4.4. External and Human-Gated Control

InstructGPT demonstrates the high-cost external path in which humans supply demonstrations and rank model outputs for later training [104]. OmniOPD demonstrates selective escalation by sending the student's peak-entropy reasoning forks to teacher verification instead of requesting feedback uniformly [105]. These designs motivate reserving external tools or humans for low-coverage or high-consequence decisions: calibrating ambiguous slices, validating a proposed policy change,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

adjudicating regressions, or authorizing promotion. Independence is thereby also a budgeted feedback-allocation problem.

A trained multimodal fact verifier offers a specialized channel for evidence-grounded claims, although its use as an adaptive orchestration gate remains to be established [106]. On reasoning and planning tasks, sound external verification has produced gains where unsupported iterative self-critique can collapse [107]. Yet external feedback still has a support boundary: symbolic checks may be silent for open-ended quality, while human overseers can inherit the system's framing and confirmation bias [219]. Governance must retain signal source, scope, and confidence so that "external" is not mistaken for universally correct.

4.5. Routing Decisions Across Stages and Model Types

4.5.1. Stage-Specific Data Destinations

The same diagnosed gap can be routed to different mutable objects. In pre-training, ADO changes domain mixture weights while training proceeds [90]. In SFT, error-driven multimodal tuning retrieves instructions aligned with diagnosed reasoning failures [13]. In preference optimization, factually augmented RLHF changes grounded comparison data [108]. In on-policy distillation (OPD), OmniOPD selects student-conditioned reasoning forks for teacher feedback [105]. In context and memory adaptation, Frontis-MA1 selects experience cards and updates a search board rather than model weights [16]. These destinations differ in update cost, latency, persistence, and the kind of failure they can repair, so stage choice is itself a data-allocation decision.

BigBang routes held-out capability gaps toward new task distributions and synthesis strategies [15], whereas MED-VRAG iteratively revises queries and accumulates page-image evidence in runtime memory [109]. These routes are alternatives, not a mandatory pipeline. Routing to a convenient stage rather than the causal bottleneck can conceal a failure temporarily, so the controller should state why its destination should alter the diagnosed mechanism and which evaluation would falsify that expectation. Existing surveys organize most self-improvement methods within a preselected component or stage, leaving matched-budget evidence for choosing among pre-training, SFT, preference data, and runtime memory sparse [213].

4.5.2. Modality-Specific Evidence and Data Granularity

OmniOPD selects uncertainty-bearing reasoning forks as textual chunk-level supervision units [105], while MED-VRAG keeps retrieved evidence as document-page images [109]. These examples show that the control functions need not change across model types, but the signal space, reliability boundary, and mutable data unit do. A controller must preserve links to spans, image regions, frames, audio segments, renders, or actions when selecting and rewriting data; a fluent textual correction can otherwise erase the evidence that made the example useful.

Existing systems illustrate several such data units. Factually augmented RLHF makes externally grounded multimodal feedback part of preference construction rather than relying on response style alone [108]. DriveStack-VLA supplies a distinct embodied example: it uses a render teacher to align supervision with bird's-eye-view geometry, so the selected correction targets a perceptual representation rather than only a textual answer [110]. In both cases, the orchestration implication is that the controller must route a diagnosed error to a data object whose modality preserves the missing evidence.

VLM judges can retain useful pairwise ranking while remaining unreliable for absolute score thresholds, making their role in a data gate use-dependent [74]. Recursive multimodal training also exhibits modality-specific collapse, and mitigation depends on which modality and label channel is refreshed [111]. Modality-aware verification and mixture constraints are therefore needed; an unsupported signal should trigger abstention or escalation rather than a confident update.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

4.6. Evidence Standards and Auditability

DataOrchestra trains an input-conditioned planner but does not report online updates to that deployed policy [14]. BigBang instead uses accumulated post-training evidence to revise later task distributions, critic criteria, and synthesis strategies [15]. This contrast shows why the presence of a teacher, judge, critic, planner, or agent is not by itself evidence of adaptive orchestration. A fixed accept/reject verdict implements only stage-local gating; the stronger claim begins when accumulated feedback changes a later target, operator, allocation, gate, or policy in a way distinguishable from the fixed rule.

Datasheets establishes a structured record of dataset motivation, composition, collection, and use [4]. The Data Provenance Initiative broadens that record to sources, creators, licensing conditions, and downstream reuse [5]. Building on these documentation precedents, we propose that an orchestration study expose five linked objects:

1. **Signal and provenance:** the observation received by the controller, its source, uncertainty, support, and prior exposure;
2. **Decision space:** the feasible alternatives for z_t, o_t, π_t, b_t, and g_t, including abstention and stopping;
3. **Data delta:** the exact records, weights, order, metadata, or memory state changed by the selected action;
4. **Decision evidence:** a comparison with a fixed, random, or ablated policy that isolates the value of adaptive selection; and
5. **Trajectory and governance:** downstream outcomes across rounds, protected-slice regressions, costs, promotions, rollbacks, and termination reasons.

Implicit Execution Tracing shows a complementary controller-side problem: when ordinary logs are absent, even segment boundaries and agent attribution must be reconstructed from embedded signals [112]. The five objects above form our proposed replayable record linking the observed signal, selected action a_t, realized data delta, and resulting learner state; reporting only a final dataset, score, or controller prompt hides the causal data delta.

DataComp supplies a controlled-comparison precedent by holding training recipe and compute scale fixed while changing the candidate multimodal dataset [1]. Adaptive orchestration requires an additional shift from static artifact documentation to intervention provenance, in which the triggering signal, feasible alternatives, selected data delta, promotion, and rollback remain linked across rounds.

Controller-level lineage is equally necessary. Automated multi-agent failure attribution asks which component caused a task failure and at which point in the trajectory [113]. Behavioral-reproducibility work adds whether repeated invocations choose the same tools, in the same order, with the same arguments [114]. Together with execution tracing above, these results motivate recording provenance by design rather than inferring it after a dataset or checkpoint has already been promoted.

DataEvolver illustrates the preparation-loop boundary: downstream SFT validates its prepared output but does not control the reported inner preparation loop [3]. DataMaster illustrates the operation-attribution boundary: downstream gains compare complete data-engineering branches but do not isolate every internal operation [98]. Arena analyses illustrate the audit-independence boundary by documenting incentives to adapt repeatedly to one selection environment [85]. Thus final-only evaluation supports pipeline adaptation plus downstream validation, jointly changed variables support system-level rather than selector-level claims, and a reused selection benchmark is not an independent audit. Each mechanism should state these boundaries locally rather than delegating them to a separate limitations discussion.

4.7. From Orchestration to Execution

BigBang makes the separation concrete: held-out evaluation identifies capability gaps, its controller revises the task and synthesis distribution, and training realizes the selected intervention [15]. Evaluation therefore establishes what has been observed; orchestration chooses a data transition; and

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

execution realizes it. DataEvolver marks an earlier boundary, closing feedback at preparation-policy refinement while using downstream training only for validation [3]. Section 5 instantiates these decisions across pre-training, SFT, reinforcement learning, on-policy distillation, and context or memory adaptation; across all stages, feedback counts as orchestration only when it changes a later data state whose effect is evaluated again.

5. Execution Mechanisms: Data-Centric Updates Across Training and Adaptation

This section turns to the execution layer: the concrete stage-specific updates through which selected feedback changes the data-related objects available to later rounds. We organize execution mechanisms by training and adaptation stage: pre-training (PT), supervised fine-tuning (SFT), reinforcement learning (RL), on-policy distillation (OPD), and context or memory adaptation. This stage-based view is useful because the same feedback signal implies different interventions at different points in the pipeline.

Table 3. Representative execution-layer mechanisms in pre-training and supervised fine-tuning. Each row records how evaluation feedback updates a mutable data-related object.

Method	Stage	Data operation	Updated object	Feedback signal	Update and loop closure	Main risk
Ultra-FineWeb [115]	PT	Corpus filtering/selection	Seed set and filter	Downstream validation	Revises seeds and filtering rules for later retained documents	Proxy overfitting; narrowed coverage
MATES [88]	PT	Model-aware selection	Pre-training examples	Current-model influence	Refreshes sample selection for later training	Early-model bias against long-tail data
Group-MATES [96]	PT	Group-level selection	Example groups	Group influence	Selects groups conditional on earlier selected data	Unstable complementarity estimates
DataEvolve [2]	PT	Rewriting/ refinement	Cleaning strategy and corpus	Judge scores and diagnostics	Revises cleaning strategies across rounds	Evaluator bias; semantic drift
ADO [90]	PT	Mixing/ reweighting	Domain sampling weights	Online loss trajectories	Updates domain proportions during training	Short-term gain bias
RegMix-D [130]	PT	Dynamic mixing	Domain mixture weights	Mixture-switch losses	Revises later data proportions from proxy trajectories	Accumulated allocation bias
DEITA [131]	SFT	Instruction selection	Instruction pool	Quality, complexity, diversity	Selects examples for SFT	Proxy-metric overfitting
LESS [12]	SFT	Target-aware selection	Instruction examples	Target-task influence	Selects examples for targeted SFT	Target-distribution overfitting
VisionFoundry [138]	SFT	Generation/ filtering	Image-instruction pairs	Verifier accept/reject	Removes unsupported pairs before SFT	Shared visual blind spots
Self-Evolving Visual Questioner [139]	SFT	Iterative generation/rewriting	Question pool	Judge verdicts and rewrites	Updated VLM generates the next pool	Evaluator entanglement
MobileGen [150]	SFT	Failure-driven repair	GUI instructions and trajectories	Capability profile	Changes next-round difficulty distribution	Incorrect failure attribution
AutoVQA-G [102]	SFT	Grounded annotation repair	VQA annotations and prompts	Critiques of failed annotations	Revises prompts for the next data round	Shared perception errors
Socratic-Geo [152]	SFT	Failure-driven synthesis	Geometry examples	Solver failures	Uses failures for targeted augmentation	Overfocus on observed failures

5.1. Pre-Training: Evaluation-Triggered Data Interventions

Pre-training is the earliest execution stage at which evaluation feedback can alter foundational training data and thereby influence subsequent model capabilities. Because the downstream effects of such interventions are often observed only after substantial training, data updates frequently rely on incomplete or early-stage evaluation signals. We treat a method as core evidence of data-centric recursive improvement when evaluation outcomes change pre-training data that are reused in a later training round; one-shot scoring and fixed corpus construction instead provide supporting evidence

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

for the corresponding data operations. According to the executed data operation, we organize related work into four families: *Corpus filtering and selection*, *data rewriting and refinement*, *data synthesis and targeted mining*, and *data mixing and reweighting*.

5.1.1. Corpus filtering and selection.

Corpus filtering and selection determine which existing examples are retained for subsequent pre-training. DataComp-LM [97] and FineWeb [6]compare alternative filtering and corpus-construction strategies through controlled pre-training, while QuRating [220] and Meta-rater [221] introduce model-generated or model-validated quality signals to guide document selection. PDS [216] and BETR [222] further estimate the training value of candidate data using training dynamics, data influence, diversity, or relevance to target tasks, while related work extends these selection mechanisms to training-loss-based pruning [223], multilingual quality filtering [224], and multimodal pre-training data curation [225]. Although these methods shift data selection away from fixed heuristics toward evaluation-informed decisions, the final training set is typically determined before the target training run begins.

More direct recursive updates arise when evaluation outcomes are used to modify subsequent data selection. Ultra-FineWeb [115] validates candidate high-quality data through downstream model performance and uses the resulting evidence to revise its seed set and filtering classifier, thereby changing which web documents are retained. GREATS [89] directly estimates candidate utility through the expected validation-loss reduction after updating the current model with each candidate, and reselects the samples used for the next parameter update at every iteration; although its large-scale experiments focus on fine-tuning, the same online-selection mechanism is also validated in a GPT-Small pre-training setting on OpenWebText.

Influence on the current model provides another direct criterion for data selection. MATES [88] locally probes the current pre-training model to estimate data influence and uses these estimates to update data selection for the next training stage. Group-MATES [96] extends this estimation to interactions among groups of examples. Layer-Aware Influence [226] uses the current batch as a self-influence reference during pre-training, estimates sample influence online from output-layer gradients, and removes negatively influential samples before each update, so that the subset actually used for training changes with the model state. Data value can also depend more directly on the current training state. BLADE [116] updates online selection according to the loss gap between candidate data and a dynamic reference that evolves with training, whereas OPUS [117] estimates candidate utility in the optimizer-induced update space, incorporating the current optimizer state into the effective update geometry, and uses the resulting projected utility to guide subsequent sampling. Data value therefore becomes not only an intrinsic property of a sample, but also a quantity that changes with model state and training stage.

However, repeated reliance on current-model feedback can narrow corpus coverage by systematically reducing exposure to long-tail, low-resource, or poorly understood data that are undervalued early in training.

5.1.2. Data Rewriting and Refinement.

Data rewriting and refinement modify the content, structure, or representation of existing training examples so that data with useful information but quality defects can remain in pre-training. ProX [118], RefineX [120], REWIRE [227], and SwallowCode and SwallowMath [228] improve existing text through example-specific processing programs, editing strategies, and generative rewriting, while AITQE [229] and PMC-InterCPT [230] extend similar operations to image–text consistency and multimodal context reconstruction. These studies show that rewriting and refinement can improve pre-training corpora, although their processing strategies are typically specified before large-scale training begins.

Building on this foundation, some methods use quality assessments of rewritten outputs to adjust the data-processing procedure itself, allowing evaluation outcomes to directly affect the content that ultimately enters pre-training. RePro [121] constructs rewards from the quality and semantic

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

faithfulness of rewritten text and uses them to optimize a rephraser, so that the updated policy produces a different rewritten pre-training corpus. CTRL [119] iteratively generates revised texts, filters out candidates that degrade readability or helpfulness, and retains lower-perplexity ones for subsequent rounds of refinement; although it targets safety-oriented (query, response) pre-training texts rather than general web corpora, the iterative revision loop illustrates the same evaluation-driven rewriting pattern.

Other methods use verification of execution results to drive further data revision. DataOrchestra [14] revises example-level processing plans according to the actual outputs of data operations. For noise pruning, a verifier checks whether useful content has been mistakenly removed; for LLM rewriting, it checks whether noise has been removed without introducing factual errors or information loss. Failed rewrites are re-executed with corrective instructions, allowing execution feedback to directly affect the final processed data. DataEvolve [2] repeatedly performs quality-problem identification, candidate strategy generation, execution, and evaluation, and uses the outcomes of one round to revise the cleaning and refinement strategies applied in the next.

Nevertheless, repeated refinement can accumulate evaluator bias, leading to semantic drift, information loss, or stylistic homogenization in the resulting corpus.

5.1.3. Data Synthesis and Targeted Mining.

Data synthesis and targeted mining both expand the current corpus by adding new training units, but they differ in where the added data come from. The former constructs new training content from existing corpora, model generations, or structured transformations, whereas the latter identifies and introduces relevant examples from external data sources according to specific needs. MIND [122], BeyondWeb [123], and FinePhrase [231] expand pre-training corpora by generating or restructuring existing content and evaluate alternative construction strategies through downstream training outcomes. DoPAMine [232] retrieves domain-relevant documents from large web corpora using domain seeds, while WRAP++ [124] exploits cross-document relations to discover and organize new training content. These approaches can effectively expand the training corpus, but their generation or acquisition targets are typically specified in advance and do not continually adapt to the current model state. Related mechanisms extend such synthesis to underrepresented modalities, for example synthetic interleaved speech–text data for pre-training [233].

Building on these approaches, some methods begin to adjust newly added data according to the behavior or learning state of the current model. Self-Improving Pretraining [125] evaluates current-model rollouts, original suffixes, and externally rewritten suffixes for quality, safety, and factuality, and uses these judgments to select the continuations used in subsequent pre-training. Active Recap Learning [234] uses the current model's long–short gap to identify tokens that depend strongly on distant context, retrieves the preceding segments most useful for predicting those tokens, and converts them into recap-augmented sequences for long-context continued pre-training. BoLT [126] iterates between latent-thought inference and model training, using latent thoughts inferred by the current model to construct new thought-augmented training data, after which the updated model participates in the next round of data construction. SynPro [127] further updates rephrasing and reformatting generators using quality, faithfulness, and data-influence rewards, and then uses the updated generation policies to produce synthetic tokens for the next training stage. Together, these methods make the selection, construction, or generation of new data increasingly dependent on the current model state and evaluation outcomes.

A related risk is that data expansion driven by model state or evaluation signals can amplify existing errors and leave knowledge regions outside the current evaluation coverage underrepresented.

5.1.4. Data Mixing and Reweighting.

Data mixing and reweighting primarily adjust the training proportions and sampling probabilities of data from different domains, sources, languages, or modalities without changing the content of individual examples. DoReMi [7], DoGE [128], Data Mixing Laws [8], and RegMix [235] use proxy

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

training, gradient relations, or mixture-performance prediction to identify improved data mixtures. CLIMB [236] further iterates between proxy training and performance prediction to progressively refine candidate mixtures over semantic data clusters. Although this search process is iterative, in all of these approaches the selected mixture is fixed before the subsequent target-model training run. More recent work instead treats the sampling distribution as a mutable data state that can be updated continually as training progresses.

Some methods dynamically adjust subsequent data proportions according to training loss and learning progress. ODM [129] updates domain sampling probabilities using recent loss reduction, while ADO [90] fits online loss scaling curves for individual domains and modifies the mixture according to the marginal gains expected from allocating additional data. Velocitune [237] and GRAPE [81] redistribute training proportions using different measures of learning progress. Aioli [238] instead fits a unified optimization model to the observed training dynamics and periodically updates the mixture accordingly, while the online variant of RegMix-D [130] uses losses observed at mixture-switch points during target training to revise subsequent data proportions.

Other methods use gradient or influence information to estimate the value of different data sources for updating the current model. Dynamic Gradient Alignment [239] adjusts sampling proportions according to gradient alignment between source domains and target data. PiKE [240] uses gradient norm and variance to modify the contributions of different domains in subsequent batches, while TiKMiX [241] uses group influence to estimate the marginal effect of increasing the proportion of a domain on validation objectives and updates the subsequent mixture accordingly.

Some approaches further combine multiple training signals to update the subsequent sampling distribution. HDS [92] integrates data quality, cross-domain influence, and model state into a multi-objective reward to update domain sampling proportions online. OP-Mix [242] instead trains lightweight domain-specific LoRA adapters from the current model, interpolates these adapters to approximate candidate data mixtures, and evaluates the resulting proxies to select the mixture used for subsequent training. AC-ODM [243] and Data Mixing Agent [244] similarly learn domain-reweighting policies from actor-critic optimization or historical data-mixing trajectories.

A corresponding risk is that dynamic mixing can overallocate training to domains with larger short-term measured gains, allowing allocation bias to accumulate across subsequent updates.

Overall, the four families above are not data operations introduced by Data-Centric Recursive Improvement itself. The key change is that data decisions traditionally fixed before training are increasingly revised according to model evaluations and training outcomes. Pre-training execution therefore shifts from one-shot data preparation toward evaluation-conditioned data updates, while repeated feedback may also lead to narrower data coverage, distorted data content, and accumulated bias in the training distribution.

5.2. Supervised Finetuning: Evaluation-Guided Instruction Data Construction

Supervised fine-tuning adapts pretrained models to task-following behavior through updates on instruction data. At this stage, the mutable object is the instruction pool, and evaluation supports recursive improvement only when outputs from the current model or an evaluator change the pool used in a later update. Evaluation takes on a control role when these outputs determine which examples are generated, selected, repaired, or grounded; a full model–data loop further requires the updated model to change the candidates or distribution in the next round.

5.2.1. Instruction Selection as a Data-Update Operator

Instruction selection treats the SFT pool as a mutable distribution. Selection criteria assess sample difficulty or complexity, diversity and coverage, and downstream utility, typically producing scalar metric scores; preference-based methods additionally use preference comparisons. A fixed selection or weighting rule maps these outputs to retention, pruning, weighting, or resampling, providing direct feedback-to-data evidence. Iterative control requires outputs from the current model or evaluator to revise the instruction pool used in a later round.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

DEITA combines quality and complexity scores with diversity-aware selection [131]. LESS estimates target-task utility through influence scores [12]. Data Diversity Matters provides supporting evidence that diversity and coverage affect robustness [93].

For multimodal data, PreSel uses an Instruction Relevance Score to assign task-wise sampling budgets and Neighbor Centrality scores to select representative images before instruction generation [80]. TIVE combines task-difficulty and instance-influence scores [17]. ScalSelect derives importance scores from instruction-conditioned attention representations and statistical leverage scores [132]. ICONS aggregates influence estimates across target tasks into a consensus for data selection [18]. COINCIDE ranks data by concept-skill transferability [95]. Align2LLaVA uses human preference comparisons to train reward models whose scalar scores filter questions and answers, after which the inner LLM rewrites and reviews the retained instructions [133].

Most of these methods implement stage-local execution rules rather than repeated model-dependent selection. The main risk is mixture overfitting to a proxy metric: selected examples may improve benchmark scores while narrowing coverage of difficult visual cases.

5.2.2. Instruction Generation and Rewriting

Instruction generation and rewriting treat instructions, responses, rationales, demonstrations, and visual dialogues as mutable SFT objects. Teachers supply candidate supervision, judges or verifiers produce scores and accept/reject verdicts, and critics provide targeted feedback or revision requests. These outputs provide direct control-role evidence when they trigger generation, filtering, or rewriting, and iterative evidence when the updated model generates the next instruction pool.

Self-Instruct generates instruction triples and maps validity and similarity checks to accept/reject decisions [134]. WizardLM uses Evol-Instruct to rewrite seed instructions into more complex variants [135]. X-InstructBLIP transfers instruction tuning across modalities [136], while Video-LLaVA extends instruction following to unified image and video inputs [137]. These methods establish reusable data operations or supervision interfaces, but do not show evaluation after a model update revising a later supervision distribution.

VisionFoundry provides direct feedback-to-data evidence through verifier-filtered construction. It generates questions, reference answers, and text-to-image prompts and then renders the corresponding images [138]. A VLM verifier assesses whether each image provides sufficient visual support for the intended answer and returns an accept/reject verdict as an actionable evaluation output. A fixed action-selection rule removes unsupported instruction–image pairs from the SFT pool, making this a one-pass execution pipeline rather than an iterative model-dependent loop.

Self-Evolving Visual Questioner provides iterative control evidence. The current VLM generates image-grounded questions, while a frozen reference checkpoint acts as a judge/evaluator and produces accept/reject verdicts and rewrite requests as actionable evaluation outputs [139]. These outputs drive action selection by retaining, removing, or revising candidate questions, and the retained examples support question-generation and question-answering objectives. After SFT, the updated model generates the next instruction pool, so evaluation plays a control role in closing the model–data feedback loop.

The main risk is evaluator entanglement: a generator, teacher, or evaluator may share the language priors or perceptual blind spots of the model being trained. Iterative-improvement claims should therefore report the evaluation output, the data update it triggered, and performance on held-out grounding checks independent of the filtering rule.

5.2.3. Grounded and Structured Supervision

This family applies grounding-to-filter to both region-level evidence and its structured OCR, layout, chart, and temporal special cases. The mutable objects include grounded QA, region captions, masks, boxes, OCR instructions, chart QA, and frame-aware video instructions. Their primary quality property is conditional correctness, while generated charts also require artifact fidelity. Depending on the evidence type, evaluation produces rule-verifiable judgments such as IoU or execution success,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

accept/reject verdicts, localization scores, or learned-evaluator rationales. Fixed acceptance, repair, annotation, or weighting rules map these outputs to SFT data updates.

Osprey introduces mask-conditioned visual instruction tuning [140], and Draw-and-Understand supports points, boxes, and free-form visual prompts [141]. LLaVAR adds OCR-aware visual instructions for text-rich images [142]. ChartGen exposes executable chart structure through code-guided generation [143]. VideoRefer Suite makes spatial-temporal reference and localization part of the instruction object [144], while Chain-of-Frames associates reasoning with frame-level evidence [145]. These methods make supervision inspectable, but the evidence remains supporting unless a grounding output controls acceptance, repair, or a later data round.

VeriEvol provides direct feedback-to-data evidence. A type-aware evolution module rewrites low-difficulty image–question seeds into harder image-grounded prompts, while the HTV-Agent evaluates the reliability of each candidate answer [146]. Independent solution traces and programmatic checks produce rule-verifiable judgments, and direct visual inspection produces an additional grounding verdict. Their combined accept/reject output drives a fixed acceptance rule that retains a candidate only when multi-source counter-evidence fails to refute its answer. The cycle updates prompt difficulty and the conditional correctness of answers before verified examples enter SFT.

The main risk is evaluator entanglement: a learned evaluator may share the perception errors of the generator. Generic QA transformations may also discard conditioning evidence such as OCR layout, numerical relations, or event order. Claims of iterative control should therefore identify the evaluation output, the resulting data update, and whether a later round depends on the updated model.

5.2.4. Failure-Driven Instruction Repair

This family converts observed failures into retrieved repair samples, hard examples, negative instructions, or revised sampling weights. The relevant evaluation outputs include error categories, missing-capability diagnoses, hard-case subsets, vulnerable subgroups, and capability profiles. These outputs support failure-directed data construction when they determine the supervision used in the next update; repeated evaluation after that update allows the repair distribution to evolve with the model.

Error-driven LMM tuning evaluates a student model and uses a teacher to produce missing-capability diagnoses that guide targeted retrieval from a task-agnostic instruction collection [13]. ARDS uses perturbation-based robustness evaluation to identify vulnerable subgroups and map their difficulty to a robustness-aware training mixture [147]. Learning from Reasoning Failures uses frontier models to analyze reasoning errors, propose targeted examples, and filter them for quality before instruction tuning [148]. LRV-Instruction instead derives positive and negative supervision from a predefined hallucination taxonomy rather than a diagnosis of the current model [149]. These methods connect diagnosis to data construction, but do not report repeated evaluation after the model update.

MobileGen implements an iterative model–data loop by repeatedly evaluating a GUI agent and producing a capability profile over structural and semantic difficulty [150]. The capability profile is a measurement output; when it determines the difficulty distribution for next-round instructions and trajectories, evaluation takes on a control role. Re-evaluating the updated agent then changes the subsequent data distribution.

Anchor Evolution converts model failures into truth-anchored SFT data through an iterative SFT–RL cycle [151]. Ground-truth answer matching produces rule-verifiable accept/reject verdicts, which are aggregated into rollout success counts and mapped by a fixed data-routing rule to redundant, volatile, and failing-frontier subsets. For the failing frontier, a teacher performs error attribution and produces error categories, retrieval keywords, and actionable hints as diagnostic outputs. The keywords retrieve verified anchors, while the hints construct scaffold-augmented supervision; after the model update, re-evaluation determines the next-round failure frontier, so evaluation progresses from measurement and diagnosis to a control role.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Socratic-Geo dynamically couples data synthesis with model learning: a Teacher agent generates geometric reasoning data, the Solver learns from it, and Solver failures guide targeted augmentation in the next round [152]. AutoVQA-G uses a consistency evaluator to critique failed annotations, after which a prompt-optimization agent revises the generation prompts for the next VQA-G data round [102]. The former updates both the model and subsequent data, whereas the latter primarily evolves the data-generation process.

RSIBench-Data provides an agent-controlled example of iterative SFT data synthesis. A researcher agent proposes message-format examples, executable tasks, or tool-use trajectories, while a fixed sandboxed evaluator returns selection scores, verifier outcomes, task trajectories, and execution diagnostics [84]. These evaluation outputs guide revisions to data sources, filtering, curriculum, mixture, and supervision format before a shared SFT backend trains the next candidate checkpoint. Because each candidate is trained from the same base model, the loop adapts the data-synthesis strategy rather than carrying model parameters forward across rounds.

The main risk is incorrect error attribution: an observed failure may result from model capacity, prompt design, evaluator bias, or contamination rather than missing instruction data. Claims of repeated improvement should therefore identify the diagnostic output, the resulting data update, and how the updated model changes the next repair distribution.

The four intervention families provide a common description of SFT-stage data updates. A method belongs to the SFT execution layer when an output from a teacher, judge, or verifier directly changes instruction examples or mixture weights. The reliability of that output is analyzed in Section 3, while a controller that chooses the evaluator, target stage, retry policy, or stopping condition belongs to the orchestration layer in Section 4. Static instruction tuning remains important background, but a full co-evolution signal requires a traceable path from evaluation output to the next supervised data object and from the updated model to a later data distribution.

5.3. Reinforcement Learning and Preference Optimization

RL and preference optimization support policy-level recursive improvement when evaluation signals change the data used in later optimization rounds. From a data-centric perspective, this process contains two linked pipelines. The policy-training pipeline constructs preference pairs, collects rollouts, and refreshes them from successive policy checkpoints. Its data are defined both by their supervision form, such as a preference pair or verifier-labeled trajectory, and by the policy that generated their candidates. The evaluator-training pipeline instead constructs supervision for a reward model, critic, process reward model, or judge that will assess later policy data. RLHF [104], DPO [245], and GRPO [156] provide representative optimization mechanisms that consume these data objects. With previous Section 3.1 defining the underlying feedback forms, particularly preference comparisons, verifier labels, and learned-judge outputs, this section shifts the focus from what those signals represent to how they construct, label, and refresh data across policy updates.

5.3.1. Feedback-Guided Policy-Training Data Generation and Refresh

Preference-pair construction and revision.

The data-centric question for preference optimization is not whether a method uses preference data, but whether feedback changes the construction rule or contents of the preference corpus across rounds. The mutable data usually take the form of chosen–rejected response pairs, sometimes augmented with localized corrections, counterfactual inputs, or task-specific annotations. A single DPO round on a fixed preference corpus may improve alignment, but it does not close a recursive loop because the preference data are not updated [245].

RLHF-V [39] is a clear representative of recursive preference-data construction. Segment-level human corrections identify localized visual errors, convert them into corrective preference pairs, and return those pairs to preference optimization. The recursive object is therefore the preference-pair corpus, not only the policy parameters. LLaVA-RLHF [108] similarly maps response comparisons and factual augmentation into policy updates through a reward model, while mDPO [246] uses image-

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Table 4. Representative execution-layer mechanisms in RL, on-policy distillation, and context or memory. The final column distinguishes standard on-policy or cross-interaction adaptation from methods that revise future data construction or management. OPD and memory adaptation do not by themselves imply a strict recursive loop.

Method	Stage	Data operation	Updated object	Feedback signal	Update mechanism	Adaptation depth / limit
Self-Rewarding LMs [47]	RL	Self-rewarded preferences	Responses and preference pairs	Model-as-judge scores	Regenerates judged pairs and applies iterative DPO	Iterative data–policy loop
On-policy LVLM [153]	RL	Preference refresh	Current-policy responses and pairs	Hallucination labels	Resamples, relabels, reweights, and optimizes each round	Iterative on-policy refresh
RLAIF-V [48]	RL	Iterative preference construction	AI-labeled response pairs	Claim-level self-feedback	Rebuilds preference data for iterative DPO	Iterative feedback–data loop
Visual-RFT [32]	RL	Verifier-scored rollouts	Current-policy rollout groups	Accuracy or IoU	Scores rollout groups and updates the policy with GRPO	Standard on-policy update
eva [155]	RL	Prompt evolution	Training prompts and responses	Creator–solver reward	Prioritizes and generates prompts for later policy updates	Prompt–policy co-adaptation
Ornith-1.5 [157]	RL	Joint data generation	Tasks, scaffolds, and rollouts	Validity, difficulty, novelty, and success	Regenerates and jointly optimizes all three with GRPO	Bounded meta-level loop
GKD [160]	OPD	Student-rollout generation	Student sequences and teacher targets	Teacher token distributions	Resamples student states for generalized distillation	On-policy adaptation with fixed collection rule
AsyncOPD [161]	OPD	Stale-rollout correction	Cached rollouts and teacher scores	Learner-time KL and cache support	Corrects old-policy data under the current student	Stale-data correction with fixed updater
TrOPD [162]	OPD	Feedback validation	Student trajectories and outlier regions	Teacher reliability under distribution shift	Masks, clips, or redirects unreliable supervision	Current-round validation with fixed rule
OmniOPD [105]	OPD	Selective supervision	High-uncertainty chunks	Teacher responses and semantic similarity	Audits uncertain forks and distills without logits	Selective supervision with fixed scheduler
OPCD [166]	OPD	Context-conditioned feedback	Student rollouts and privileged context	Context-conditioned teacher distributions	Distills contextual knowledge on student states	On-policy adaptation with fixed context
Flux-OPD [167]	OPD	Evolving context	Student rollouts and contexts	Context difference and conflict	Revises contextual correction and later supervision	Cross-round context refresh
Reflexion [171]	Mem.	Reflection memory	Verbal reflections	Environment outcomes	Writes failure lessons for later trials	Across-trial accumulation with fixed writer
AgentRR [172]	Mem.	Verified replay	Structured experience traces	Task outcome and replay checks	Abstracts, verifies, and reuses traces on later tasks	Cross-task replay with fixed replay rule
Voyager [173]	Mem.	Skill accumulation	Executable skill library	Execution and self-verification	Stores verified skills for later composition	Skill accumulation with fixed manager
SelfMem [175]	Mem.	Strategy refinement	Memory strategy	Downstream task feedback	Explores and revises storage and retrieval behavior	Strategy-level adaptation
EvolveMem [176]	Mem.	Retrieval co-evolution	Knowledge and retrieval configuration	Failures, regression, and stagnation	Diagnoses failures, revises configuration, and rolls back	Retrieval-policy co-evolution
Frontis-MA1 [16]	Mem.	Experience reuse	Program nodes, cards, and task board	Sandbox outcome, progress, and novelty	Selects parents, retrieves traces, and promotes verified data	Evaluated reuse with fixed selection weights

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

conditioned comparisons to improve text–image alignment. These methods differ in the source of contrast, but they share the same data operation: feedback defines which responses should be paired and how the preference relation should be interpreted.

Preference-data recursion can also be policy-conditioned. On-policy LVLM alignment [153] samples hallucinated responses from the current policy, annotates them with preferences, and uses the resulting pairs to optimize that same policy. After the policy update, the next candidate pool is sampled from a changed model, so the preference corpus evolves with the policy. TangoFlux [154] provides a related example in audio generation. It repeatedly generates audio candidates, ranks them with a fixed CLAP evaluator, and rebuilds preference pairs for subsequent optimization. Because CLAP remains fixed, the recursive update occurs in the policy-training data rather than in the evaluator-training data.

The main risk is recursive preference misspecification. If constructed pairs encode annotator bias or superficial proxies such as style, verbosity, or formatting, later rounds can amplify these shortcuts and reduce coverage of rare failure modes. Preference-data recursion therefore requires explicit criteria, counterfactual pairs, coverage across failure types, held-out agreement checks, and provenance linking each comparison to its evidence and evaluator.

Verifier-labeled rollout generation, selection, and weighting.

A second policy-training pathway uses verifier feedback to select or weight policy-generated trajectories. The mutable object is a trajectory or rollout group, and the feedback may come from an exact-match answer, task metric, executable checker, or rule-based verifier. In this pathway, the verifier is treated as a fixed feedback source. Recursion arises only when its outputs change which rollouts enter the policy update or how strongly they are weighted. A verifier used only for final reporting remains an ordinary evaluation mechanism.

Visual-RFT [32] is a representative case. It turns classification accuracy or IoU from a reporting metric into feedback for scored perception trajectories. MM-Eureka [33] provides a related example for image reasoning, where rule-based feedback is applied to current-policy rollouts. In both cases, the verifier does not merely measure performance; it changes the effective training set by filtering or weighting policy-generated data.

When verifier-scored rollouts are collected on policy, the loop becomes deeper because the policy that generates the data is also updated. GRPO alternates rollout collection with policy optimization, so each checkpoint changes the generation distribution for the next batch [156]. The recursive object is therefore not only the selected rollout set, but also the data-generating policy. Ornith-1.5 [157] serves as a representative model that incorporates such process into model post-training. It generates tasks, task-specific scaffolds, and solution rollouts jointly, and optimizes all three with GRPO using task validity, frontier difficulty, novelty, scaffold alignment, reward fidelity, hack resistance, and rollout success. Because frontier difficulty depends on current-policy performance and stronger policies generate the tasks and scaffolds used in later rounds, evaluation refreshes both the rollout distribution and part of its generation configuration. However, the outer reward definitions, environment, and GRPO update rule remain fixed, so the method represents bounded meta-level self-improvement rather than open-ended self-referential RSI. This creates a data-freshness requirement: rollout labels and policy versions should be tracked so that updates are based on data generated by a compatible policy.

The main risk is verifier narrowing. Because verifier labels decide which rollouts are reinforced, the policy may overfit measurable signals while neglecting unmeasured qualities such as grounding, consistency, plausibility, or safety. This risk grows when the verifier is treated as a complete proxy for task quality.

5.3.2. Learned Evaluator Data Construction and Updating

The learned-evaluator pathway updates the supervision used to train a reward model, critic, process reward model, or judge. Its mutable object is a preference annotation, rubric, or intermediate-step label, not the rollout set itself. Unlike verifier-guided rollout selection, which changes rollout

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

inclusion or weights, this pathway changes the evaluator that will later label, rank, or weight policy data. It is recursive only when evaluation revises the evaluator's training data, rubric dimensions, or supervision structure; a fixed evaluator used only for filtering belongs to rollout selection or rejection sampling.

Omni-RRM [75] is a clear closed-loop example. It synthesizes rubric-grounded omni-modal preference supervision, trains an evaluator on that supervision, and uses the evaluator to assess later policy data. The recursive object is therefore not only the policy-data pool, but also the evaluator-training corpus that defines future judgments. VisionReward [158] provides complementary evidence for learned-evaluator construction by building a multidimensional reward model from fine-grained human judgments over image and video candidates.

Evaluator supervision can also move from whole-output judgments to structured intermediate labels. VisualPRM [52] trains process-level supervision for visual reasoning. This changes the granularity of the feedback object: the unit of supervision becomes an intermediate reasoning step rather than a final response. If these labels are updated using later model failures, the evaluator can reshape which reasoning errors are detected and corrected in subsequent policy training.

The main risks are axis collapse and evaluator entanglement. Aggregate rewards may hide underrepresented quality dimensions, and evaluators trained on same-family model outputs or judgments can become self-confirming. Recursive use therefore requires calibration, external validation, disagreement checks, and clear provenance for evaluator supervision.

Overall, these pathways differ in recursive depth. Fixed preference data and a fixed reward model support only a one-shot policy update. Recursion begins when feedback changes later preference data, rollout selection or weights, evaluator supervision, or the policy that generates future candidates. A method approaches policy-level self-improvement only when feedback also changes the configuration of future improvement, such as what data to collect, which failures to target, which evaluator to trust, or which update operator to apply. Thus, we treat an RL method as co-evolutionary when evaluation feedback changes a concrete data object, and as recursively improving when it also changes the rules governing future data collection, labeling, weighting, or optimization.

5.4. On-Policy Distillation from Teacher Feedback

On-policy distillation (OPD) supports teacher-mediated self-improvement by training on states visited by the current student and supervising those states with teacher feedback. Its data-centric feature is that the supervision data are mutable: as the student changes, the rollout-feedback records used for training also change. Each record links a student-visited state or prefix to the produced action or token, the teacher signal, and the decision about how that signal enters the update.

OPD differs from SFT in Section 5.2 because it does not rely only on a fixed instruction pool, and from RL in Section 5.3 because it usually provides denser feedback than scalar rewards or preferences. In this section, we ask whether teacher feedback only updates the student, changes the rollout-feedback records used in later updates, or also changes how future supervision is collected and constructed. We organize the discussion around two pipelines: the student-side rollout pipeline and the teacher-feedback pipeline, following the OPD survey [247] while focusing on mutable data objects rather than only feedback or loss types.

5.4.1. Student-Conditioned Rollout Generation and Validation

Generating supervision on current-student states.

The student-side pipeline starts by sampling trajectories from the current student and attaching teacher feedback to the states or prefixes that the student actually visits. This is the basic data-centric operation in OPD: the supervised states are generated by the current student rather than drawn only from a fixed instruction pool.

MiniLLM [159] and GKD [160] instantiate this operation by combining student-sampled sequences with teacher token distributions and reverse-divergence updates. GKD can also mix on-policy and fixed-data distillation. These methods make the training data follow the changing student, but they

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

do not by themselves establish recursive improvement because feedback does not revise the rule for collecting future supervision.

Correcting stale or unsupported rollout feedback.

A rollout-feedback pair may become unreliable when it is stale, weakly supported by the teacher, or located in an outlying teacher–student region. The data-centric operation is to decide whether that pair should be used directly, corrected, downweighted, constrained, or excluded. This is the OPD form of the signal-validity issue discussed in Section 3.

AsyncOPD [161] addresses staleness by treating cached rollouts as mutable records. Each record stores visited prefixes, sampled actions, behavior-policy information, teacher scores, and the student version. At update time, the learner recomputes the relevant quantities under the current student and applies old-to-current correction instead of treating the cached estimate as fixed. This controls temporal mismatch between rollout collection and student update, but it does not test teacher reliability on every visited state.

Trust Region OPD [162] addresses teacher–student mismatch. It distills only where teacher supervision is considered reliable and constrains updates so that outlier regions do not dominate learning. Together, these methods show record-level data co-evolution: evaluation changes how existing rollout-feedback pairs are used, but not the teacher's context or the policy for constructing future feedback.

5.4.2. Teacher-Feedback Construction and Adaptation

Selecting teacher access and supervision granularity.

The teacher-feedback pipeline can change the feedback record even when on-policy sampling is fixed. It can choose which part of the rollout receives supervision and which teacher interface supplies the signal.

Existing methods instantiate this choice at different granularities. PRISM [163] uses response-level supervision from a discriminator and does not require teacher logits. One-Token Rollout [164] applies feedback at a single supervised prefix. OmniOPD [105] selects uncertain chunks and performs logit-free chunk-level distillation across models with different tokenizers. OPOD [165] routes student responses to specialist teachers and controls their influence separately.

These methods fit the data-centric view because they change how feedback records are constructed. They become recursive only if outcomes revise later teacher routing, supervision granularity, or teacher-influence policies. If those rules stay fixed, the methods adapt supervision but do not yet adapt the supervision policy itself.

Conditioning feedback on supervisory context.

Teacher feedback can also be conditioned on context that the student does not see. In this setting, the context is part of the feedback record rather than part of the student's sampling input. If the context is fixed, it changes the teacher signal for the current update. If the context is selected or revised by feedback, it becomes part of the supervision policy.

On-Policy Context Distillation (OPCD) [166] samples student trajectories without privileged context and evaluates each token with a teacher conditioned on historical solution traces or an optimized system prompt. It then uses reverse-KL distillation to transfer this contextual knowledge into the student. During training, however, the context for each input remains fixed. OPCD therefore adapts the student responses, but it does not revise the supervisory context itself.

Flux-OPD [167] makes the context mutable. It extracts contexts from recent student behavior, compares context-conditioned and context-free teacher feedback, and downweights conflicting context corrections. Each round changes both the student rollouts and the context set used for later supervision. This is closer to recursive improvement because student behavior revises part of the future feedback-construction process, not only the next training examples.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Grounding feedback and assigning update credit.

Dense teacher feedback must be grounded in task-relevant evidence. Vision-OPD [248] does this by letting a crop-conditioned self-teacher observe an answer-critical region and provide token-level feedback for a student rollout from the full image. Visual-Advantage OPD [249] goes further by comparing token scores under the original image and a visually degraded image, using the difference as an evidence-dependent update weight.

The same student-visited-state view extends beyond text prefixes to denoising states, video histories, and audio-conditioned rollouts [250–253]. In the mechanisms reviewed here, the evidence signal mainly grounds or weights the current update rather than revising future evidence selection, teacher access, or student sampling.

5.4.3. When OPD Becomes Teacher-Mediated Recursive Improvement

The OPD loop is adaptive because the current student generates the rollouts used for the next update. It becomes record-level data co-evolution when evaluation changes how rollout-feedback records are corrected, filtered, or weighted. It approaches recursive self-improvement only when feedback also changes the future supervision policy, such as teacher routing, context construction, evidence selection, or supervision granularity.

This boundary is important for validation. Rollouts may become stale, teacher feedback may fail under teacher–student mismatch, contexts may conflict, and teacher-like fluency may improve without stronger use of task evidence. Freshness ablations, mismatch analysis, context-stability tests, evidence counterfactuals, and equal-compute comparisons with SFT or RL are therefore needed. OPD should not be counted as recursive improvement merely because it uses teacher feedback; the feedback must change how future supervision is collected or constructed.

5.5. Context and Memory as Data-Centric Adaptation

Context and memory support *experience-level self-improvement* by changing the evidence and prior behavior available to future interactions, even when model weights remain fixed. Their mutable data includes retrieved evidence, runtime context, persistent records, replayable traces, and reusable skills compiled from validated experience. The signals defined in Section 3, including relevance, grounding, environment outcomes, freshness, contradiction, and compression fidelity, govern how these objects are selected, compressed, written, revised, retained, replayed, compiled into skills, promoted, or deleted. We organize these interventions into two linked paths. The fast, non-parametric path selects and adapts runtime evidence and context, updates persistent memory, and verifies and reuses prior experience without changing model weights. The slower, weight-updating path promotes verified memory into SFT or RL data. Under our action-based boundary, either path is co-evolutionary only when evaluation changes the data available to a later interaction or training round, and it approaches recursive self-improvement when feedback also revises the strategy governing that change.

5.5.1. Fast Non-Parametric Experience Adaptation
Formulating Memory Through Selection, Compression, and Persistent Updates

The fast path first determines which evidence enters the runtime context and which information persists across interactions. Its mutable objects include retrieval rankings, over-budget contexts, and stored records, while recall, attribution, relevance, freshness, contradiction, task utility, and compression fidelity govern query reformulation, reranking, retention, merging, summarization, consolidation, revision, and deletion. These quantities are evaluation signals of the kind defined in Section 3.1, while the policy mapping them to later memory actions becomes adaptive orchestration only when feedback revises that policy. RAG [254], RETRO [255], VisRAG [256], and ColPali [257] establish retrieval over external or structured evidence but do not themselves close a co-evolutionary loop because answer evaluation does not change later retrieval. Iterative RAG updates accumulated evidence within an interaction according to retrieval relevance, answer quality, and attribution [109], and MIRAGE takes a related step for visual understanding by iterating retrieval and answer revision,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

although such methods need not revise the retriever for future tasks [258]. As evidence accumulates, context systems manage its growth through sparse video memory in MovieChat [259], online memory banks in MA-LMM [260], and tool-queryable state in VideoAgent [261], while evaluation-guided compression methods select or merge evidence under task-quality and resource constraints, as in visual context compression [262], VideoChat-Flash [263], and AVOC [264]. Persistent-memory systems extend this management across interactions by moving records between stores in MemGPT [168], consolidating interactions in Mem0 [169], organizing linked notes in A-Mem [265], or reconstructing explicit state in ReaSMoRy [170]. Whether these mechanisms preserve useful experience without excessive context growth is evaluated over extended or dynamic interactions through retention and temporal consistency in LoCoMo [266], long-memory retrieval in LongMemEval [267], evolving-memory quality in EvoMemory [268], beyond-million-token stress tests [269], and stored-record quality, downstream utility, and error propagation in memory-management evaluation [270].

Using Memory During Inference

Once formulated, memory affects later behavior by retrieving and replaying evaluated trajectories, critiques, workflows, distilled lessons, or executable skills. Reflexion converts task experience into verbal reflections [171], ExpeL distills reusable knowledge from experience [271], Agent Record & Replay verifies and replays prior traces under replay-time validity [172], and M^2 compresses and retrieves prior traces according to abstraction quality or task relevance [272]. Skills extend this mechanism from recalling experience to retrieving procedures that directly structure future behavior. Voyager stores self-verified executable programs [173], MUSE-Autoskill refines retrieved skills using unit tests and runtime feedback [174], and SAGE uses a skill-integrated reward to guide skill generation and reuse [273]. These methods accumulate system-level capability without necessarily changing model weights, and they become co-evolutionary when replay outcomes revise which experiences or skills are retained, updated, downweighted, or deleted.

Adapting the Memory-Management System

A stronger loop changes not only memory content but also the strategy governing storage, retrieval, and reuse. Frontis-MA1/OpenMLE illustrates the boundary between content-level and strategy-level adaptation by converting sandbox outcomes into experience cards and a task-level board that guide parent selection and bounded operation-conditioned context construction [16]. However, it retains every deterministic card and applies fixed selection weights, providing stronger evidence for evaluation-guided memory use than for co-evolution of the memory policy itself. SelfMem moves beyond fixed management rules by allowing an agent to evaluate and refine its memory strategy through memory tools and conversational feedback [175]. EvolveMem makes this adaptation more structured by exposing retrieval configuration as an action space and using failure diagnosis, regression-triggered reversion, and stagnation-triggered exploration to co-evolve stored knowledge and retrieval [176]. BigBang broadens the adaptation target from memory management to the overall improvement strategy. Its generator harness records the motivation, outcome, failure cause, and conclusion of each data-synthesis experiment, while critic and held-out evaluation feedback revise the synthesis program and evaluation criteria in later rounds [15]. These records therefore support adaptive orchestration of the improvement strategy rather than direct learning of the memory policy, consistent with Section 4.6. Across these systems, learned deletion, contradiction-aware consolidation, versioned rollback, and transfer-aware skill deprecation remain underexplored despite their importance for preventing stale, conflicting, or harmful experience from accumulating.

5.5.2. Slow Weight-Updating Adaptation through Memory Promotion

Constructing Training Data From Verified Experience

Unlike the fast path, which changes the data used in later interactions without necessarily updating model weights, the slow path promotes verified runtime records into SFT or RL data, making the candidate trace pool and promoted training subset its mutable objects. Each candidate trace retains

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

the observation or evidence, model action, tool state, outcome, and provenance needed for audit. An evaluator checks its utility, grounding, freshness, and robustness under replay or counterfactual tests, after which admission and normalization convert it into an SFT instruction or demonstration, as discussed in Section 5.2, or an RL preference pair or scored rollout, as discussed in Section 5.3. After training, held-out evaluation determines whether the promoted data improved the intended capability and whether the source traces or derived examples should be retained, reweighted, or removed in the next round, coupling the fast interaction-memory loop to a slower cycle of data construction, weight updates, and newly generated experience. Several works instantiate this promotion pathway at different levels of experience granularity. STaR records self-generated rationales and promotes those that yield correct answers into iterative SFT data [177]. ReST applies the same generate-evaluate-promote principle to offline policy improvement by reusing policy-generated and reward-evaluated responses [178]. In executable software engineering, SWE-Gym uses unit tests to select successful agent trajectories for iterative rejection-sampling fine-tuning [179]. Agent-R complements success-only selection by converting erroneous interactive trajectories into reflection data, identifying the first error and connecting it to a corrective search branch [180]. Frontis-MA1 broadens this pathway to long-horizon program search by converting sandbox-verified trajectories into execution-grounded SFT and RL data [16].

Evaluating Memory-Derived Training Corpora

Memory promotion reintroduces the signal contamination, judge bias, and attribution risks discussed in Section 3. Evaluator independence is crucial because a plausible error can become both memory and supervision when the same model generates, approves, stores, and trains on a trace. Safeguards include replay-time verification, as illustrated by Agent Record & Replay [172], as well as executable task outcomes, human review, separately calibrated verifiers, and held-out grounded evaluation. Each derived training example should also preserve a provenance link to its source trace and supporting evidence. Beyond evaluator contamination, memory-derived corpora can inherit stale or irrelevant records, lossy removal of decisive evidence, and invalid transfer of earlier experience. Independent auditing and traceable provenance instantiate the signal-independence safeguard discussed in Section 4.4 and determine whether promoted experience provides reliable supervision across stages. Although this promotion pathway remains less mature than runtime retrieval and replay, it connects memory adaptation to data-evaluation co-evolution.

Table 5 summarizes representative methods under the proposed Evaluation–Orchestration–Execution taxonomy, coding each work by the feedback signals it consumes, the orchestration decisions it makes, and the mutable data-related objects it updates.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Table 5. Representative data–evaluation co-evolution methods positioned along the Evaluation–Orchestration–Execution taxonomy. A filled bullet (●) marks dimensions where the method, as described in this survey, actively uses feedback to change a later data-related state; an open circle (○) marks dimensions that are absent, fixed, or used mainly for validation rather than loop control.

	Evaluation signal					Orchestration decision				Execution target				
Method	**Score**	**Failure**	**Verifier**	**Pref./Judge**	**Env.**	**Signal**	**Attrib.**	**Action**	**Gate**	**PT**	**SFT**	**RL/Pref.**	**Eval. data**	**Memory**
DataComp [1]	●	○	○	○	○	○	○	●	○	●	○	○	○	○
GREATS [89]	●	○	○	○	○	●	○	●	○	●	○	○	○	○
ICONS [18]	●	○	○	○	○	●	○	●	○	○	●	○	○	○
Ultra-FineWeb [115]	●	○	●	○	○	●	○	●	●	●	○	○	○	○
MATES [88]	●	○	○	○	○	●	○	●	○	●	○	○	○	○
DataOrchestra [14]	○	○	●	●	○	●	●	●	●	●	○	○	○	○
DataMaster [98]	●	●	○	●	○	●	●	●	●	●	●	○	○	●
DataEvolver [3]	○	●	●	○	○	●	●	●	●	●	○	○	○	●
DataEvolve [2]	●	●	○	●	○	●	●	●	●	●	○	○	○	●
BigBang [15]	●	●	●	●	○	●	●	●	●	○	●	○	●	●
Frontis-MA1 [16]	●	●	●	○	●	●	●	●	●	○	●	●	○	●
Multimodal DataEvolver [100]	○	●	●	●	○	●	●	●	●	○	●	○	○	●
VisionFoundry [138]	○	○	●	○	○	●	○	●	●	○	●	○	○	○
Self-Evolving Visual Questioner [139]	○	●	●	●	○	●	●	●	●	○	●	○	○	○
Visual-RFT [32]	●	○	●	○	○	●	○	●	●	○	○	●	○	○
MM-Eureka [33]	●	○	●	○	○	●	○	●	●	○	○	●	○	○
RLHF-V [39]	○	●	○	●	○	●	●	●	●	○	○	●	●	○
RLAIF-V [48]	○	●	○	●	○	●	●	●	●	○	○	●	●	○
Omni-RRM [75]	○	○	○	●	○	●	○	●	●	○	○	●	●	○
AsyncOPD [161]	●	○	○	●	○	●	○	●	●	○	○	○	○	●
Flux-OPD [167]	●	○	○	●	○	●	○	●	●	○	○	○	○	●
SelfMem [175]	●	●	○	●	●	●	●	●	●	○	○	○	○	●
EvolveMem [176]	●	●	○	●	●	●	●	●	●	○	○	○	○	●
SWE-Gym [179]	●	●	●	○	●	●	●	●	●	○	●	●	○	●
Agent-R [180]	●	●	●	●	●	●	●	●	●	○	●	●	○	●
Voyager [173]	○	●	●	○	●	●	●	●	●	○	○	○	○	●
MUSE-Autoskill [174]	○	●	●	○	●	●	●	●	●	○	○	○	○	●

Abbreviations. Score: benchmark, downstream, or validation score. Failure: failure cluster or error diagnosis. Verifier: rule, tool, execution, IoU, exact-match, or fixed-evaluator signal. Pref./Judge: preference, reward-model, learned-judge, critique, or rationale signal. Env.: environment outcome or executable interaction result. Signal: signal selection. Attrib.: error attribution. Action: target, operator, allocation, or budget decision. Gate: acceptance, stopping, rollback, or promotion gate. PT: pre-training. RL/Pref.: reinforcement learning or preference optimization. Eval. data: evaluator-supervision data. Memory: runtime context, persistent memory, experience records, or skill stores. The table codes functional roles as described in this survey rather than implementation details. A method may contain additional components; a filled bullet is used only when the component participates in the feedback-to-update path emphasized here.

6. Failure Modes of Data–Evaluation Co-Evolution

A higher score is not sufficient evidence that a data–evaluation loop has improved. The score may measure the wrong capability, reuse information seen by the loop, hide unstable updates, overlook lost data support, or combine changes that cannot be attributed to one component. We use five requirements: signal validity and coverage, signal independence, update stability, data-support integrity, and system identifiability. Reproducible traces are evidence needed to establish system identifiability. These requirements are our synthesis of the surveyed evidence. They are separate diagnostic questions rather than claims that every recursive-improvement system must fail.

6.1. Invalid or Incomplete Feedback Signals

A feedback signal is invalid when it does not measure the capability that the loop is intended to improve. Limited target-space coverage creates the same problem because an accurate score on a narrow test may not represent the full target distribution. Repeated updates can then improve the measured proxy while leaving the intended capability unchanged. Multimodal evaluation makes this gap especially important: wide task coverage, low cost, and low variance are difficult to achieve together [181].

Grounding failure is a pointwise form of invalidity. A policy may answer through a language shortcut instead of using the relevant image, audio, video, document, spatial state, or action signal. The evaluator may miss the same shortcut. Visual encoders can overlook fine detail [24], and strong vision–language models still fail perceptual tasks designed to resist memorized language patterns [182]. Task-specific evidence also shows that automatic evaluators can remain biased when text and images provide different information [183].

Aggregate reliability does not resolve this problem. A judge can return repeatable scores while remaining insensitive to required evidence. Because each judgment can decide which example enters the next update, pointwise validity matters more than leaderboard consistency alone. The unresolved

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

boundary is whether the signal responds to the intended capability across the relevant domains and modalities.

6.2. Dependent or Exposed Feedback Signals

A signal can measure the right capability and still fail to provide independent evidence. This occurs when evaluation examples have influenced training, when the same benchmark is reused for repeated selection, or when the generator and evaluator share correlated preferences. In each case, the loop can improve against information that is already part of the update process.

Contamination is difficult to rule out. If a model has seen only the training split, a train–test perplexity contrast may reveal exposure. If it has seen both splits, both values may fall together and resemble a clean model [187]. Other probability- and membership-based detectors also depend on assumptions that can fail across domains and training regimes [188]. Release-filtered and dynamic benchmarks reduce some exposure channels, but they can have smaller samples and higher measurement variance [190].

Generator–evaluator dependence creates a related problem without direct item leakage. Evaluators can recognize and favor outputs from their own model family [189]. Self-preference is not proof that the selected output is worse, and current evidence does not isolate it as the cause of every downstream regression. The limitation is that a gain measured by a related evaluator may not persist under independent examples, judges, or external outcomes.

6.3. Unstable Updates and Non-Monotonic Progress

Update instability occurs when an accepted or deployed state regresses or oscillates under a fixed, independent evaluation. Variation among candidates that were never accepted is normal search behavior and is not itself instability. The risk arises when discrete changes to prompts, memories, workflows, code, or data are accepted using textual critique or noisy judge scores that do not provide a reliable update direction. Intrinsic self-critique can reduce performance where sound external verification improves it [107], and self-refinement can strengthen a model's preference for its own outputs without improving the intended metric [193].

The balance between corrected and introduced errors provides one direct stability test. A recent preprint models this balance with an error-correction rate and an error-introduction rate. Several tested models degraded or oscillated when correction did not offset newly introduced errors [194]. This result is task- and model-specific, but it shows why an update that fixes some failures may still reduce total performance.

Search reports must separate candidate exploration from accepted-state performance. The Darwin Gödel Machine finds strong coding agents through lineages that sometimes include lower-scoring descendants [195]. These lower-scoring descendants are useful exploration, not evidence that the deployed system regressed. The example shows why candidate scores and the best stored score should be reported separately. A recent self-evolving-agent preprint reports that an unguarded context update can regress, while a disjoint held-out gate can reject the measured regression [83]. Such gates protect the selection split; they do not guarantee true out-of-distribution improvement.

6.4. Data-Support Loss under Recursive Generation

Data-support loss occurs when the update policy repeatedly removes rare but relevant parts of the training distribution. Recursive generation can copy model errors and narrow the range of later training data [198]. Average reward or fluency may rise while rare classes, styles, failure cases, or out-of-distribution behaviors disappear. This loss can remain hidden behind a higher task score. Iterative SFT, DPO, and mixed SFT–DPO can raise pass@1 while diversity and out-of-distribution performance decline [199]. Maximum-likelihood sharpening can concentrate probability mass without adding knowledge [274], and curated feedback can improve expected reward while variance falls [275]. Variance collapse is therefore one possible symptom of support loss, not a common cause of all five requirements.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

The outcome depends on how generated data is used. Replacing real data with recursively generated data collapsed in analyzed settings, while retaining real data across rounds avoided collapse in the corresponding accumulation setting [82]. Asymptotic analyses retain a more pessimistic boundary: repeated recursion can distort the learned distribution even when each round contains only a small synthetic fraction [200].

The same support question applies across modalities. If text examples are easier to generate, verify, or optimize, a mixed data policy may expand text supervision faster than image, audio, video, spatial, or action supervision. Cross-modal recursion may also behave differently when one modality is regenerated and another remains fixed [111]. Current evidence does not establish a universally safe real-to-synthetic ratio or modality mixture. The boundary depends on the generator, filtering policy, model, domain, accumulation rule, and number of rounds.

6.5. Unattributable and Irreproducible System Changes

The behavior of a co-evolution system is jointly determined by the base model, controller, judge, prompts, tools, external APIs, and environment state. Reports that retain only the final score cannot identify which component or interaction caused a change. System identifiability requires localizing that effect through controlled component changes and replay. This is component attribution: it asks which part of the running system produced the change. It differs from training-data attribution, which asks which examples caused a model behavior and which data intervention may repair it.

Judge biases are well documented [203], but multi-agent systems still lack a detailed account of why their control processes fail [103]. Automated component attribution is only beginning to be studied [113]. Reconstructing an execution path after only the final output remains partial and costly [112].

Replayable state is a prerequisite for this attribution. The record must include model and API versions, prompts, tool schemas, retrieval state, rejected candidates, and intermediate controller decisions. Otherwise a component swap may change both the target component and hidden execution conditions. Stochastic inference adds another source of variation across steps. One recent preprint reports that tool sequences are more repeatable than their argument values and that much behavioral divergence begins early in the pipeline [114]. This evidence is preliminary, but it shows why a final model, dataset, or score is insufficient for replay. The unresolved limitation is whether a reported gain can be localized to a component or interaction under a controlled, replayable configuration.

7. Future Directions

Section 6 identified five requirements that a measured gain can fail to satisfy. This section presents five corresponding research programs. Some programs directly address current failures. Others develop the multimodal data, feedback, and control mechanisms needed for future co-evolution systems. The common goal is to establish when a loop produces reliable improvement rather than a higher score on its own update signal.

7.1. Build Valid, Coverage-Aware Multimodal Signals

Any signal that selects data or changes a model should be validated at the level where it is used. Aggregate model rankings are not enough because a single incorrect judgment can change which example enters the next update. Evaluation should therefore test consistency, target coverage, grounding, and uncertainty separately. A repeatable judge can still measure the wrong capability, and one score can hide conflicting changes in factuality, visual quality, safety, or temporal coherence.

Multimodal signals require direct grounding tests. A judge may reward a fluent answer without using the relevant image, audio, video, or document. Current multimodal judges vary across tasks and can miss changes in hallucination, spatial reasoning, factual grounding, and visual fidelity [74,184,276]. Recent reward benchmarks cover multimodal and interleaved inputs, but process-level validation remains limited [185,210].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Evidence ablation provides one practical test. The evaluator can be run again after the required modality or premise is masked. If its judgment does not change, the signal needs further validation before it is used for training [184,186]. Premise-level checks can also record which visual facts support a reward [77]. Repeated scoring can estimate variance. Low-confidence judgments should retain their uncertainty instead of being converted directly into training labels.

Future evaluators should also report separate dimensions rather than one aggregate score. For generation, these dimensions may include prompt alignment, factuality, visual quality, identity, physical plausibility, safety, and temporal coherence. For embodied tasks, they may include outcome correctness, use of the required modality, valid tool use, and recovery after failure. Each dimension should be checked against human judgments and held-out domains.

Unified understanding–generation models may help construct these tests. An editing model can change one visual fact and create a counterfactual pair for an evaluator. Understanding errors can also become generation tasks, and generated counterexamples can be used to test understanding [277–280]. These counterfactuals should then pass the independent-validation controls in Section 7.2.

7.2. Establish and Scale Independent Feedback

A valid signal is not sufficient if it has already influenced training or shares the model's preferences. Future loops need feedback that is separated from data generation, update selection, or model training. This separation can be provided by private examples, unrelated judges, executable rules, human review, or outcomes from an external environment.

Dynamic and private evaluation can reduce repeated exposure. Release-date filtering [190], continuous refresh [19], and generated instances [36] reduce direct reuse of fixed test items [10]. They also introduce costs. Release filtering can reduce sample size, and generated tests can share assumptions with the model being evaluated. Reports should therefore state who generated each test, who verified it, which components share training data, and how much uncertainty remains.

Independent evaluators provide another check. Cross-family judges, tools, and humans may reduce correlated errors, but model family alone does not establish independence. Evaluation should measure agreement and error correlation on held-out examples. Evaluator disagreement should be recorded and handled by the escalation policy in Section 7.3.

Rule-verifiable tasks provide machine-checkable feedback. A curriculum agent and executor can co-evolve from no external task data [191], and a generator and verifier can co-evolve to reduce repeated consensus errors [101]. Scoring rules and test cases can also change with the system they evaluate [192,281]. These methods are most useful when the verifier is based on an external rule, program, or environment transition rather than another unconstrained model judgment.

Real interaction provides a different source of evidence. People, deployed software, and the physical world can supply corrections, interruptions, preferences, task outcomes, environment transitions, and recovery attempts. Multimodal interaction models and embodied agents may lower the cost of collecting these signals [282–284]. Interactive benchmarks capture part of this process [55–57], but they do not replace long-term feedback from real deployment.

Future collection systems should retain the context, outcome, and uncertainty of each interaction. They should report useful feedback per unit cost, human burden, privacy risk, safety controls, interaction diversity, and transfer to new users and environments. Simulation can supplement this evidence, but it should not be reported as independent real-world feedback.

7.3. Measure and Control Update Stability

Current work does not provide a reliable rule for choosing the number of feedback rounds or rollouts per round. The canonical self-rewarding method reports three iterations and leaves longer runs open [47]. The solver–verifier capability gap can limit self-improvement [196], reward-filtered fine-tuning can improve and then saturate [197], and weak external signal can increase variance [285]. These results identify relevant variables, but they do not provide a general scaling law.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

7.3.1. Track Trajectories

Future studies should report the full update trajectory. Each round should include task performance, proposed candidates, accepted states, the best state found so far, rollout count, verifier acceptance rate, reversals, rollbacks, and cost. This separates actual round behavior from a best-score curve that is non-decreasing by construction. Support and out-of-distribution metrics should be reported separately under the data-support program in Section 7.4.

Stability should be measured directly. For refinement, reports can compare the number of corrected errors with the number of new errors [194]. For search, they can report candidate rejection, reversal, and rollback rates. The same candidates should be evaluated under stable criteria across rounds so that changes in the judge are not mistaken for changes in the model.

7.3.2. Treat Indistinguishable States as Ties

Near a benchmark ceiling, small score differences require special care. Repeated evaluation, confidence intervals, and paired held-out tests can show whether two states are distinguishable. If the difference is within evaluation noise, the controller should treat the result as a tie. The loop should then stop, collect more evidence, or move to a harder independent test.

7.3.3. Predefine Escalation and Rollback

Human escalation and rollback should use explicit triggers. Useful triggers include high evaluator uncertainty, disagreement between independent judges, repeated reversals, falling diversity, held-out regression, weak transfer, or a safety alarm. Model judges and human reviewers both have systematic errors [219,286]. Debate can help in some settings, but it can also optimize for judge preference [287,288]. The escalation policy should therefore state which signal caused the handoff and what evidence the human or stronger evaluator must check.

7.4. Scale Data while Preserving Support

Sample count alone does not define data scale. The update policy should preserve rare capabilities, difficult cases, modalities, and out-of-distribution behavior across rounds. This requires direct comparison of replacement, accumulation, and curation policies over several real-to-synthetic ratios.

Fresh real data remains important. Accumulating real and generated data can behave differently from replacing real data with generated data [82]. Provenance records should identify the generator, source, round, filtering rule, and whether a sample was accumulated or replaced. Current provenance infrastructure remains incomplete [5,289]. The real-to-synthetic ratio should be reported as both a data policy and a cost.

Several mechanisms can increase on-policy multimodal data. Multimodal reinforcement learning with verifiable rewards (RLVR) uses rule-checked tasks, as shown by TRON for visual reasoning and VideoRLVR for spatial and temporal trajectories [201,202]. Executable artifacts provide another check: Video2Code converts interaction videos into webpages that can be tested on state transitions [290].

Generated and learned environments can provide resettable state and parallel rollouts. Agent World Model and ScaleEnv generate tool-use environments [291,292], while VLAW combines real-robot data, a video world model, synthetic rollouts, and policy updates [293]. These routes differ in realism and verification, but they share one unresolved question: whether more generated experience preserves capability, modality, tail, and out-of-distribution coverage.

Environment selection matters as much as environment count. Adding more multimodal environment types can cause negative transfer, while ability-aware selection and curricula can perform better than training on the full pool [294]. Reports should include capability coverage, difficulty, temporal horizon, modality use, accepted verified rollouts per attempt, verifier confidence, and held-out transfer. They should also record whether the agent exploited the verifier or ignored a required modality [295,296].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

7.5. Localize and Reproduce System Changes

7.5.1. Localize the System Change

A loop should first identify which running component or interaction caused a change. Studies can swap or ablate the model, controller, judge, prompts, tools, APIs, and environment, then test whether the same effect appears during replay. Automated component attribution and trace reconstruction provide initial tools for this task [112,113]. The result should reproduce under a matched execution state before it is used to guide a repair.

7.5.2. Choose a Repair After Localization

Training-data attribution can then ask which examples contributed to the behavior and which data change may repair it. Text methods can select data based on predicted task effects [87] or identify harmful examples before repair [204]. Influence methods scale to language models, although their identified examples are not always reliable [297,298]. Fixed, off-policy targets may not represent errors produced after an update. On-policy attribution can reduce this mismatch [91]. The repair can target a data or training stage, or a runtime component such as memory, tools, workflow, or environment. Stage-selection policies should be compared with fixed-stage baselines under the same diagnosed failure. A state-distribution view can compare offline SFT with RL and on-policy distillation on states produced by the current learner [299], while freshness controls can reduce updates from stale rollouts [300]. The repair is useful only if it improves held-out behavior without reducing retention or safety.

7.5.3. Compare Under Controlled Conditions

Controlled testbeds are needed to compare these choices. They should allow planned component changes while recording model versions, prompts, data, evaluators, tools, rounds, and budgets. Every loop should also be compared with a compute-matched static baseline. The baseline should spend the same budget without repeated feedback, for example on more seed data, more epochs, one generation batch, or one-shot distillation. This comparison tests whether iteration adds value. A fixed-budget comparison should include task generation, policy rollouts, verifier execution, human judgment, and fresh external data. It should report performance, transfer, coverage, wall-clock time, and component cost.

7.5.4. Keep a Replayable Record

Logs should include generated, accepted, rejected, and rolled-back candidates; evaluator outputs; prompts; tool calls; component versions; environment state; human actions; costs; and stopping decisions. These records support the study of orchestration failures and runtime governance [103,205]. The system should publish the monitored value, threshold, action, and rollback state for every stop rule.

7.5.5. Cross-Program Reporting

The five research programs above require a shared reporting protocol. A recursive-improvement claim should not report only the final score; it should also record the feedback signals, updated objects, controller decisions, candidate history, data provenance, human involvement, stopping rules, costs, and transfer tests that define the loop. Table 6 gives a recommended reporting template.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Table 6. Recommended reporting fields for a data–evaluation co-evolution loop.

Field	What to record	Purpose
Evaluation signal	Source, modality, version, verifier type, uncertainty, and validity checks	Shows what information drives the loop
Updated object	Data, model stage, rollout group, memory, tool, workflow, or environment	Locates the intervention
Controller policy	Selection rules, prompts, thresholds, and schedule	Makes update decisions reproducible
Candidate history	Proposed, accepted, rejected, and rolled-back candidates with reasons	Separates search history from the final state
Provenance and support	Generator, real-to-synthetic ratio, accumulation policy, diversity, tails, and modality balance	Records how the data distribution changes
Human involvement	Trigger, task, judgment type, decision, and cost	Supports escalation and oversight claims
Rounds and stopping	Round count, monitored values, halt rule, and rollback state	Shows why the loop continued or stopped
Cost	Task generation, rollouts, verifiers, humans, and fresh data	Enables fixed-budget comparison
Transfer tests	Held-out tasks, users, environments, and safety checks	Tests whether the update generalizes

8. Conclusions

Data-centric recursive improvement marks a shift in foundation model development from static data construction and post-hoc evaluation toward closed-loop systems in which evaluation actively shapes future data and model behavior. As foundation models are increasingly improved through synthetic data, preference feedback, verifier signals, retrieval, memory, and agent-environment interaction, understanding how evaluation signals are converted into data-centric interventions has become essential. In this survey, we provide a structured review of this trend through the lens of data–evaluation co-evolution. We organize existing work around a signal-decision-update loop: what signal diagnoses the current system, who or what decides how to act on that signal, and what data-related object is updated to affect the next round. This perspective connects methods across pre-training, supervised fine-tuning, preference optimization, on-policy distillation, context adaptation, and memory-based systems. We further highlight the risks that arise when evaluation becomes part of the improvement loop, including unreliable feedback, feedback overfitting, unstable updates, data-support loss, and irreproducible system changes. Addressing these challenges will require more grounded feedback signals, independent audits, stable update mechanisms, and reproducible records of data and control decisions. We hope this survey provides a foundation for researchers and practitioners to design, analyze, and advance more adaptive, robust, and accountable recursive foundation model improvement method that is a potential key to ASI.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gadre, S.Y.; Ilharco, G.; Fang, A.; Hayase, J.; Smyrnis, G.; Nguyen, T.; Marten, R.; Wortsman, M.; Ghosh, D.; Zhang, J.; et al. DataComp: In search of the next generation of multimodal datasets. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36. Datasets and Benchmarks Track.
2. Mi, T.; Shan, D.; Huang, Z.; Qin, Y.; Xie, M.; Qiao, Y.; Liu, Y.; Zhou, C.; Liu, P. Data Darwinism Part II: DataEvolve – AI Can Autonomously Evolve Pretraining Data Curation. *arXiv preprint arXiv:2603.14420* **2025**.
3. Deng, C.; Zhang, S.; Fan, J.; Du, X. DataEvolver: Automatic Data Preparation for Large Language Models through Multi-Level Self-Evolving, 2026, [arXiv:cs.DB/2606.07001].
4. Gebru, T.; Morgenstern, J.; Vecchione, B.; et al. Datasheets for Datasets. *Communications of the ACM* **2021**, *64*, 86–92.
5. Longpre, S.; Mahari, R.; Chen, A.; Obeng-Marnu, N.; Sileo, D.; Brannon, W.; Muennighoff, N.; Khazam, N.; Kabbara, J.; Perisetla, K.; et al. A large-scale audit of dataset licensing and attribution in AI. *Nature Machine Intelligence* **2024**, *6*, 975–987. https://doi.org/10.1038/s42256-024-00878-8.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

6. Penedo, G.; et al. FineWeb: Decanting the Web for the Finest Text Data at Scale. In Proceedings of the Advances in Neural Information Processing Systems, Datasets and Benchmarks Track, 2024, [2406.17557].
7. Xie, S.M.; et al. DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining. In Proceedings of the Advances in Neural Information Processing Systems, 2023, [2305.10429].
8. Ye, J.; Liu, P.; Sun, T.; Zhan, J.; Zhou, Y.; Qiu, X. Data Mixing Laws: Optimizing Data Mixtures by Predicting Language Modeling Performance. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.CL/2403.16952].
9. Li, J.; Lu, W.; Fei, H.; Luo, M.; Dai, M.; Xia, M.; Jin, Y.; Gan, Z.; Qi, D.; Fu, C.; et al. A Survey on Benchmarks of Multimodal Large Language Models. *arXiv preprint arXiv:2408.08632* **2024**.
10. Chen, S.; Chen, Y.; Li, Z.; Jiang, Y.; Wan, Z.; He, Y.; Ran, D.; Gu, T.; et al. Recent Advances in Large Language Model Benchmarks against Data Contamination: From Static to Dynamic Evaluation. *arXiv preprint arXiv:2502.17521* **2025**.
11. Gao, H.a.; Geng, J.; Hua, W.; Hu, M.; Juan, X.; Liu, H.; Liu, S.; Qiu, J.; Qi, X.; Ren, Q.; et al. A Survey of Self-Evolving Agents: What, When, How, and Where to Evolve on the Path to Artificial Super Intelligence. *Transactions on Machine Learning Research* **2026**.
12. Xia, M.; Malladi, S.; Gururangan, S.; Arora, S.; Chen, D. LESS: Selecting Influential Data for Targeted Instruction Tuning. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning. PMLR, 2024, Vol. 235, pp. 54104–54132, [2402.04333].
13. Yao, B.M.; Wang, Q.; Huang, L. Error-driven Data-efficient Large Multimodal Model Tuning. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics, 2025, [2412.15652].
14. Huang, Z.; Wang, Y.; Xia, S.; Liu, P. DataOrchestra: Learning to Orchestrate Per-Example Curation of Pretraining Data, 2026, [arXiv:cs.CL/2607.24717].
15. PolarSeeker. BigBang: Pursuing Open-Ended Intelligence through Self-Evolving Synthesis of Verifiable Frontier Tasks. Technical report, PolarSeeker, 2026. Preprint. Technical Report.
16. Yang, J.; Jiang, C.; Fu, Y.; Luo, T.; Ren, C.; Wang, W.; Zhao, K.; Liu, H.; Zuo, Y.; Wang, Y.; et al. Frontis-MA1: Training an AI4AI Model towards Recursive Self-Improvement in Machine Learning Engineering. *arXiv preprint arXiv:2607.28568* **2026**.
17. Liu, Z.; Zhou, K.; Zhao, W.X.; Gao, D.; Li, Y.; Wen, J.R. Less is More: High-value Data Selection for Visual Instruction Tuning. In Proceedings of the Proceedings of the 33rd ACM International Conference on Multimedia. Association for Computing Machinery, 2025, pp. 3712–3721, [2403.09559]. https://doi.org/10.1145/3746027.3755160.
18. Wu, X.; Xia, M.; Shao, R.; Deng, Z.; Koh, P.W.; Russakovsky, O. ICONS: Influence Consensus for Vision-Language Data Selection, 2024, [2501.00654].
19. White, C.; Dooley, S.; Roberts, M.; Pal, A.; Feuer, B.; Jain, S.; Shwartz-Ziv, R.; Jain, N.; Saifullah, K.; Dey, S.; et al. LiveBench: A Challenging, Contamination-Limited LLM Benchmark. In Proceedings of the International Conference on Learning Representations, 2025.
20. Shabtay, N.; Polo, F.M.; Doveh, S.; Lin, W.; Mirza, M.J.; Choshen, L.; Yurochkin, M.; Sun, Y.; Arbelle, A.; Karlinsky, L.; et al. LiveXiv – A Multi-Modal Live Benchmark Based on Arxiv Papers Content. In Proceedings of the International Conference on Learning Representations, 2025.
21. Yue, X.; Zheng, T.; Ni, Y.; Wang, Y.; Zhang, K.; Tong, S.; Sun, Y.; Yu, B.; Zhang, G.; Sun, H.; et al. MMMU-Pro: A More Robust Multi-discipline Multimodal Understanding Benchmark. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2025, pp. 15134–15186. https://doi.org/10.18653/v1/2025.acl-long.736.
22. Fu, L.; Kuang, Z.; Song, J.; Huang, M.; Yang, B.; Li, Y.; Zhu, L.; Luo, Q.; Wang, X.; Lu, H.; et al. OCRBench v2: An Improved Benchmark for Evaluating Large Multimodal Models on Visual Text Localization and Reasoning. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38, pp. 107930–107981. Datasets and Benchmarks Track, https://doi.org/10.52202/085713-3253.
23. Zheng, D.; Huang, Z.; Liu, H.; Zou, K.; He, Y.; Zhang, F.; Gu, L.; Zhang, Y.; He, J.; Zheng, W.S.; et al. VBench-2.0: Advancing Video Generation Benchmark Suite for Intrinsic Faithfulness. *arXiv preprint arXiv:2503.21755* **2025**.
24. Tong, S.; Liu, Z.; Zhai, Y.; Ma, Y.; LeCun, Y.; Xie, S. Eyes Wide Shut? Exploring the Visual Shortcomings of Multimodal LLMs. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 9568–9578.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

25. Tong, S.; Brown, E.; Wu, P.; Woo, S.; Middepogu, M.; Akula, S.C.; Yang, J.; Yang, S.; Iyer, A.; Pan, X.; et al. Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37. https://doi.org/10.52202/079017-2771.
26. Zhao, Z.; Wang, B.; Ouyang, L.; Dong, X.; Wang, J.; He, C. Beyond Hallucinations: Enhancing LVLMs through Hallucination-Aware Direct Preference Optimization. *arXiv preprint arXiv:2311.16839* **2023**.
27. Zhou, Y.; Cui, C.; Rafailov, R.; Finn, C.; Yao, H. Aligning Modalities in Vision Large Language Models via Preference Fine-tuning. *arXiv preprint arXiv:2402.11411* **2024**.
28. Zhang, R.; Jiang, D.; Zhang, Y.; Lin, H.; Guo, Z.; Qiu, P.; Zhou, A.; Lu, P.; Chang, K.W.; Qiao, Y.; et al. MATHVERSE: Does Your Multi-modal LLM Truly See the Diagrams in Visual Math Problems? In Proceedings of the Computer Vision – ECCV 2024. Springer, 2024, Vol. 8, pp. 169–186. https://doi.org/10.1007/978-3-031-73242-3_10.
29. Kiela, D.; Bartolo, M.; Nie, Y.; Kaushik, D.; Geiger, A.; Wu, Z.; Vidgen, B.; Prasad, G.; Singh, A.; Ringshia, P.; et al. Dynabench: Rethinking Benchmarking in NLP. In Proceedings of the Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2021, pp. 4110–4124. https://doi.org/10.18653/v1/2021.naacl-main.324.
30. Li, B.; Lin, Z.; Peng, W.; Nyandwi, J.d.D.; Jiang, D.; Ma, Z.; Khanuja, S.; Krishna, R.; Neubig, G.; Ramanan, D. NaturalBench: Evaluating Vision-Language Models on Natural Adversarial Samples. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37. Datasets and Benchmarks Track.
31. Li, X.L.; Kaiyom, F.; Liu, E.Z.; Mai, Y.; Liang, P.; Hashimoto, T. AutoBencher: Towards Declarative Benchmark Construction. In Proceedings of the International Conference on Learning Representations, 2025.
32. Liu, Z.; Sun, Z.; Zang, Y.; Dong, X.; Cao, Y.; Duan, H.; Lin, D.; Wang, J. Visual-RFT: Visual Reinforcement Fine-Tuning. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2025, pp. 2034–2044.
33. Meng, F.; Du, L.; Liu, Z.; Zhou, Z.; Lu, Q.; Fu, D.; Han, T.; Shi, B.; Wang, W.; He, J.; et al. MM-Eureka: Exploring the Frontiers of Multimodal Reasoning with Rule-based Reinforcement Learning. *arXiv preprint arXiv:2503.07365* **2025**.
34. Liu, J.; Liu, G.; Liang, J.; Li, Y.; Liu, J.; Wang, X.; Wan, P.; Zhang, D.; Ouyang, W. Flow-GRPO: Training Flow Matching Models via Online RL. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38, pp. 45650–45685. https://doi.org/10.52202/085713-1362.
35. Yu, Q.; Zhang, Z.; Zhu, R.; Yuan, Y.; Zuo, X.; Yue, Y.; Dai, W.; Fan, T.; Liu, G.; Liu, J.; et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38. https://doi.org/10.52202/085713-3775.
36. Zhu, K.; Chen, J.; Wang, J.; Gong, N.Z.; Yang, D.; Xie, X. DyVal: Dynamic Evaluation of Large Language Models for Reasoning Tasks. In Proceedings of the International Conference on Learning Representations, 2024.
37. Zou, C.; Guo, X.; Yang, R.; Zhang, J.; Hu, B.; Zhang, H. DynaMath: A Dynamic Visual Benchmark for Evaluating Mathematical Reasoning Robustness of Vision Language Models. In Proceedings of the International Conference on Learning Representations, 2025.
38. Zhang, J.; Huang, W.; Ma, Z.; Michel, O.; He, D.; Gupta, T.; Ma, W.C.; Farhadi, A.; Kembhavi, A.; Krishna, R. Task Me Anything. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37. Datasets and Benchmarks Track.
39. Yu, T.; Yao, Y.; Zhang, H.; He, T.; Han, Y.; Cui, G.; Hu, J.; Liu, Z.; Zheng, H.T.; Sun, M.; et al. RLHF-V: Towards Trustworthy MLLMs via Behavior Alignment from Fine-grained Correctional Human Feedback. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 13807–13816.
40. Chiang, W.L.; Zheng, L.; Sheng, Y.; Angelopoulos, A.N.; Li, T.; Li, D.; Zhu, B.; Zhang, H.; Jordan, M.I.; Gonzalez, J.E.; et al. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 8359–8388.
41. Lu, Y.; Jiang, D.; Chen, W.; Wang, W.Y.; Choi, Y.; Lin, B.Y. WildVision: Evaluating Vision-Language Models in the Wild with Human Preferences. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37. Datasets and Benchmarks Track.
42. Li, L.; Xie, Z.; Li, M.; Chen, S.; Wang, P.; Chen, L.; Yang, Y.; Wang, B.; Kong, L. Silkie: Preference Distillation for Large Visual Language Models. *arXiv preprint arXiv:2312.10665* **2023**.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

43. Wang, W.; Chen, Z.; Wang, W.; Cao, Y.; Liu, Y.; Gao, Z.; Zhu, J.; Zhu, X.; Lu, L.; Qiao, Y.; et al. Enhancing the Reasoning Ability of Multimodal Large Language Models via Mixed Preference Optimization. *arXiv preprint arXiv:2411.10442* **2024**.
44. Xu, J.; Liu, X.; Wu, Y.; Tong, Y.; Li, Q.; Ding, M.; Tang, J.; Dong, Y. ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation. *Advances in Neural Information Processing Systems* **2023**, *36*, 15903–15935.
45. Wu, X.; Hao, Y.; Sun, K.; Chen, Y.; Zhu, F.; Zhao, R.; Li, H. Human Preference Score v2: A Solid Benchmark for Evaluating Human Preferences of Text-to-Image Synthesis. *arXiv preprint arXiv:2306.09341* **2023**.
46. Wallace, B.; Dang, M.; Rafailov, R.; Zhou, L.; Lou, A.; Purushwalkam, S.; Ermon, S.; Xiong, C.; Joty, S.; Naik, N. Diffusion Model Alignment Using Direct Preference Optimization. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 8228–8238. https://doi.org/10.1109/CVPR52733.2024.00786.
47. Yuan, W.; Pang, R.Y.; Cho, K.; Li, X.; Sukhbaatar, S.; Xu, J.; Weston, J. Self-Rewarding Language Models. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 57905–57923.
48. Yu, T.; Zhang, H.; Li, Q.; Xu, Q.; Yao, Y.; Chen, D.; Lu, X.; Cui, G.; Dang, Y.; He, T.; et al. RLAIF-V: Open-Source AI Feedback Leads to Super GPT-4V Trustworthiness. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025, pp. 19985–19995. https://doi.org/10.1109/CVPR52734.2025.01861.
49. Xiong, T.; Wang, X.; Guo, D.; Ye, Q.; Fan, H.; Gu, Q.; Huang, H.; Li, C. LLaVA-Critic: Learning to Evaluate Multimodal Models. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025, pp. 13618–13628.
50. Lee, S.; Kim, S.; Park, S.H.; Kim, G.; Seo, M. Prometheus-Vision: Vision-Language Model as a Judge for Fine-Grained Evaluation. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024. Association for Computational Linguistics, 2024, pp. 11286–11315.
51. Wang, X.; Wang, P.; Pei, J.; Shen, W.; Peng, Y.; Hao, Y. Skywork-VL Reward: An Effective Reward Model for Multimodal Understanding and Reasoning. *arXiv preprint arXiv:2505.07263* **2025**.
52. Wang, W.; Gao, Z.; Chen, L.; Chen, Z.; Zhu, J.; Zhao, X.; Liu, Y.; Cao, Y.; Ye, S.; Zhu, X.; et al. VisualPRM: An Effective Process Reward Model for Multimodal Reasoning. *arXiv preprint arXiv:2503.10291* **2025**.
53. Li, L.; Wei, Y.; Xie, Z.; Yang, X.; Song, Y.; Wang, P.; An, C.; Liu, T.; Li, S.; Lin, B.Y.; et al. VL-RewardBench: A Challenging Benchmark for Vision-Language Generative Reward Models. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025, pp. 24657–24668.
54. Chen, Z.; Wen, Z.; Du, Y.; Zhou, Y.; Cui, C.; Han, S.; Weng, J.; Wang, C.; Tong, Z.; Huang, L.; et al. MJ-Bench: Is Your Multimodal Reward Model Really a Good Judge for Text-to-Image Generation? In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38. Datasets and Benchmarks Track, https://doi.org/10.52202/085713-2081.
55. Xie, T.; Zhang, D.; Chen, J.; Li, X.; Zhao, S.; Cao, R.; Hua, T.J.; Cheng, Z.; Shin, D.; Lei, F.; et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37. Datasets and Benchmarks Track.
56. Koh, J.Y.; Lo, R.; Jang, L.; Duvvur, V.; Lim, M.C.; Huang, P.Y.; Neubig, G.; Zhou, S.; Salakhutdinov, R.; Fried, D. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In Proceedings of the Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2024.
57. Rawles, C.; Clinckemaillie, S.; Chang, Y.; Waltz, J.; Lau, G.; Fair, M.; Li, A.; Bishop, W.; Li, W.; Campbell-Ajala, F.; et al. AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents. In Proceedings of the International Conference on Learning Representations, 2025.
58. Bai, H.; Zhou, Y.; Cemri, M.; Pan, J.; Suhr, A.; Levine, S.; Kumar, A. DigiRL: Training In-The-Wild Device-Control Agents with Autonomous Reinforcement Learning. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
59. Qi, Z.; Liu, X.; Iong, I.L.; Lai, H.; Sun, X.; Sun, J.; Yang, X.; Yang, Y.; Yao, S.; Xu, W.; et al. WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning. In Proceedings of the International Conference on Learning Representations, 2025.
60. Sun, Q.; Cheng, K.; Ding, Z.; Jin, C.; Wang, Y.; Xu, F.; Wu, Z.; Jia, C.; Chen, L.; Liu, Z.; et al. OS-Genesis: Automating GUI Agent Trajectory Construction via Reverse Task Synthesis. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Papers). Association for Computational Linguistics, 2025, pp. 5555–5579. https://doi.org/10.18653/v1/2025.acl-long.277.

61. Lambert, N.; Pyatkin, V.; Morrison, J.; Miranda, L.J.V.; Lin, B.Y.; Chandu, K.R.; Dziri, N.; Kumar, S.; Zick, T.; Choi, Y.; et al. RewardBench: Evaluating Reward Models for Language Modeling. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025. Association for Computational Linguistics, 2025, pp. 1755–1797.
62. Guo, D.; Yang, D.; Zhang, H.; Song, J.; Wang, P.; Zhu, Q.; Xu, R.; Zhang, R.; Ma, S.; et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* **2025**, *645*, 633–638. https://doi.org/10.1038/s41586-025-09422-z.
63. Lightman, H.; et al. Let's Verify Step by Step. In Proceedings of the International Conference on Learning Representations, 2024, [2305.20050].
64. Chen, D.; Chen, R.; Zhang, S.; Wang, Y.; Liu, Y.; Zhou, H.; Zhang, Q.; Wan, Y.; Zhou, P.; Sun, L. MLLM-as-a-Judge: Assessing Multimodal LLM-as-a-Judge with Vision-Language Benchmark. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 6562–6595.
65. Zhou, Y.; Fan, Z.; Cheng, D.; Yang, S.; Chen, Z.; Cui, C.; Wang, X.; Li, Y.; Zhang, L.; Yao, H. Calibrated Self-Rewarding Vision Language Models. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
66. Ghosh, D.; Hajishirzi, H.; Schmidt, L. GenEval: An Object-Focused Framework for Evaluating Text-to-Image Alignment. *Advances in Neural Information Processing Systems* **2023**, *36*, 52132–52152.
67. Lin, Z.; Pathak, D.; Li, B.; Li, J.; Xia, X.; Neubig, G.; Zhang, P.; Ramanan, D. Evaluating Text-to-Visual Generation with Image-to-Text Generation. In Proceedings of the Computer Vision – ECCV 2024. Springer, 2024, pp. 366–384.
68. Li, B.; Lin, Z.; Pathak, D.; Li, J.; Fei, Y.; Wu, K.; Ling, T.; Xia, X.; Zhang, P.; Neubig, G.; et al. GenAI-Bench: Evaluating and Improving Compositional Text-to-Visual Generation. *arXiv preprint arXiv:2406.13743* **2024**.
69. Kirstain, Y.; Polyak, A.; Singer, U.; Matiana, S.; Penna, J.; Levy, O. Pick-a-Pic: An Open Dataset of User Preferences for Text-to-Image Generation. *Advances in Neural Information Processing Systems* **2023**, *36*, 36652–36663.
70. Jiang, D.; Guo, Z.; Zhang, R.; Zong, Z.; Li, H.; Zhuo, L.; Yan, S.; Heng, P.A.; Li, H. T2I-R1: Reinforcing Image Generation with Collaborative Semantic-level and Token-level CoT. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38.
71. Li, Y.; Wang, H.; Zhang, Q.; Xiao, B.; Hu, C.; Wang, H.; Li, X. UniEval: Unified Holistic Evaluation for Unified Multimodal Understanding and Generation. *arXiv preprint arXiv:2505.10483* **2025**.
72. Wang, Y.; Zang, Y.; Li, H.; Jin, C.; Wang, J. Unified Reward Model for Multimodal Understanding and Generation. *arXiv preprint arXiv:2503.05236* **2025**.
73. Zhao, Y.; Zhou, L.; Xu, X.; Liu, Z.; Wu, J.; Fang, L. Fair in Mind, Fair in Action? A Synchronous Benchmark for Understanding and Generation in UMLLMs. *arXiv preprint arXiv:2603.00590* **2026**.
74. Kumar, D.; Tayebati, S.; Naik, D.; Krishnan, R.; Trivedi, A.R. VLM Judges Can Rank but Cannot Score: Task-Dependent Uncertainty in Multimodal Evaluation. *arXiv preprint arXiv:2604.25235* **2026**.
75. Kong, Z.; Ma, D.; Xu, Z.; Yang, A.; Ru, Y.; Wang, H.; Zhou, Z.; Bie, F.; Xiang, L.; Wu, H.; et al. Omni-RRM: Advancing Omni Reward Modeling via Automatic Rubric-Grounded Preference Synthesis, 2026, [arXiv:cs.CL/2602.00846].
76. Li, D.; Sun, R.; Huang, Y.; Zhong, M.; Jiang, B.; Han, J.; Zhang, X.; Wang, W.; Liu, H. Preference Leakage: A Contamination Problem in LLM-as-a-judge. In Proceedings of the International Conference on Learning Representations, 2026, [arXiv:cs.LG/2502.01534].
77. Wang, J.; Guan, D.; Qiu, W.; Li, Z.; Gai, Y.; Yang, Z.; Zhou, M.; Zhao, E.; et al. Grounding the Score: Explicit Visual Premise Verification for Reliable Vision-Language Process Reward Models. *arXiv preprint arXiv:2603.16253* **2026**.
78. Nguyen, D.; Ho, M.K.; Ta, H.; Nguyen, T.T.; Chen, Q.; Rav, K.; Dang, Q.D.; Ramchandre, S.; Phung, S.L.; Liao, Z.; et al. Localizing Before Answering: A Benchmark for Grounded Medical Visual Question Answering. In Proceedings of the Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, 2025, pp. 7670–7678. https://doi.org/10.24963/ijcai.2025/853.
79. Gao, X.; Wang, X.; Zhu, Y.; Cai, M.; He, C.; Wu, L. Closing the Data Loop: Using OpenDataArena to Engineer Superior Training Datasets. *arXiv preprint arXiv:2601.09733* **2025**.
80. Safaei, B.; Siddiqui, F.; Xu, J.; Patel, V.M.; Lo, S.Y. Filter Images First, Generate Instructions Later: Pre-Instruction Data Selection for Visual Instruction Tuning. In Proceedings of the Proceedings of the IEEE/CVF

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

Conference on Computer Vision and Pattern Recognition (CVPR), 2025, pp. 14247–14256, [2503.07591]. https://doi.org/10.1109/CVPR52734.2025.01329.

81. Fan, S.; Glarou, M.I.; Jaggi, M. GRAPE: Optimize Data Mixture for Group Robust Multi-target Adaptive Pretraining. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38, [2505.20380].
82. Gerstgrasser, M.; Schaeffer, R.; Dey, A.; Rafailov, R.; Korbak, T.; Sleight, H.; Agrawal, R.; Hughes, J.; Pai, D.B.; Gromov, A.; et al. Is Model Collapse Inevitable? Breaking the Curse of Recursion by Accumulating Real and Synthetic Data. In Proceedings of the Proceedings of the First Conference on Language Modeling, 2024.
83. Nguyen, M.; Nguyen, Q.; Vuong, P. Recursive Self-Evolving Agents via Held-Out Selection. *arXiv preprint arXiv:2606.28374* **2026**.
84. Meng, F.; Du, L.; Chen, Q.; Zhao, Z.; Lu, H.; Hu, M.; Shieh, M.Q. RSIBench-Data: Benchmarking Data-Centric Research for Recursive Self-Improvement. *arXiv preprint arXiv:2607.25886* **2026**.
85. Singh, S.; Nan, Y.; Wang, A.; D'Souza, D.; Kapoor, S.; Üstün, A.; Koyejo, S.; Deng, Y.; Longpre, S.; Smith, N.A.; et al. The Leaderboard Illusion. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38, pp. 86910–86964. https://doi.org/10.52202/085713-2620.
86. Qi, Z.; Baby, S.M.; Baby, S.A.; Yuan, K.; Tomkins, A.; Vu, T.; Juan, D.C.; Rashtchian, C. On the Generalization Gap in Self-Evolving Language Model Reasoning. *arXiv preprint arXiv:2606.01075* **2026**. ICML 2026.
87. Engstrom, L.; Feldmann, A.; Madry, A. DsDm: Model-Aware Dataset Selection with Datamodels. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 12491–12526.
88. Yu, Z.; Das, S.; Xiong, C. MATES: Model-Aware Data Selection for Efficient Pretraining with Data Influence Models. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37, [arXiv:cs.CL/2406.06046].
89. Wang, J.T.; Wu, T.; Song, D.; Mittal, P.; Jia, R. Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems* **2024**, *37*, 131197–131223.
90. Jiang, Y.; Zhou, A.; Feng, Z.; Malladi, S.; Kolter, J.Z. Adaptive Data Optimization: Dynamic Sample Selection with Scaling Laws. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.LG/2410.11820].
91. Wang, Z.; Li, L.; Yu, T.; Yao, Y. DRIFT: Refining Instruction Data via On-Policy Data Attribution. *arXiv preprint arXiv:2606.18307* **2026**.
92. Dang, C.; Ma, J.; Liao, M. Holistic Data Scheduler for LLM Pre-training via Multi-Objective Reinforcement Learning. In Proceedings of the Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1. ACM, 2026, p. 176–187. https://doi.org/10.1145/3770854.3780325.
93. Bukharin, A.; Li, S.; Wang, Z.; Yang, J.; Yin, B.; Li, X.; Zhang, C.; Zhao, T.; Jiang, H. Data Diversity Matters for Robust Instruction Tuning. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024. Association for Computational Linguistics, 2024, pp. 3411–3425, [2311.14736]. https://doi.org/10.18653/v1/2024.findings-emnlp.195.
94. Zhang, C.; Zhong, H.; Zhang, K.; Chai, C.; Wang, R.; Zhuang, X.; Bai, T.; Qiu, J.; Cao, L.; Fan, J.; et al. Harnessing Diversity for Important Data Selection in Pretraining Large Language Models. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.AI/2409.16986].
95. Lee, J.; Li, B.; Hwang, S.J. Concept-skill Transferability-based Data Selection for Large Vision-Language Models. In Proceedings of the Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2024, pp. 5060–5080, [2406.10995].
96. Yu, Z.; Peng, F.; Lei, J.; Overwijk, A.; Yih, S.; Xiong, C. Group-Level Data Selection for Efficient Pretraining. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38, [arXiv:cs.CL/2502.14709].
97. Li, J.; Fang, A.; Smyrnis, G.; Ivgi, M.; Jordan, M.; Gadre, S.; Bansal, H.; Guha, E.; Keh, S.; Arora, K.; et al. DataComp-LM: In search of the next generation of training sets for language models. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37. Datasets and Benchmarks Track, https://doi.org/10.52202/079017-0455.
98. Du, Y.; Yang, X.; Zhou, Z.; Liu, W.; Lei, Z.; Chen, Z.; Liu, F.; Wu, H.; Cai, Y.; Liu, Z.; et al. DataMaster: Data-Centric Autonomous AI Research. *arXiv preprint arXiv:2605.10906* **2026**.
99. Chen, D.; Huang, Y.; Ma, Z.; Chen, H.; Pan, X.; Ge, C.; Gao, D.; Xie, Y.; Liu, Z.; Gao, J.; et al. Data-Juicer: A One-Stop Data Processing System for Large Language Models. In Proceedings of the Companion of the 2024 International Conference on Management of Data. ACM, 2024, pp. 120–134.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

100. Yan, S.; Gao, Y.; Wang, Y.; Mao, D.; Wang, A.J. DataEvolver: Self-Evolving Multi-Agent Data Construction for Text-Rich Image Generation, 2026, [arXiv:cs.CV/2606.31537].
101. Pan, T.; Yan, Y.; Wang, Z.; Zhang, R.; Hou, G.; Zhang, W.; Lu, W.; Xiao, J.; Shen, Y. CoVerRL: Breaking the Consensus Trap in Label-Free Reasoning via Generator-Verifier Co-Evolution. In Proceedings of the Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), San Diego, California, United States, 2026; pp. 29833–29853. https://doi.org/10.18653/v1/2026.acl-long.1376.
102. Hu, R.; Guan, R.; Di, Y.; Bao, J.; Liu, Y. AutoVQA-G: Self-Improving Agentic Framework for Automated Visual Question Answering and Grounding Annotation. *arXiv preprint arXiv:2604.17488* **2026**.
103. Cemri, M.; Pan, M.Z.; Yang, S.; Agrawal, L.A.; Chopra, B.; Tiwari, R.; Keutzer, K.; Parameswaran, A.G.; Klein, D.; Ramchandran, K.; et al. Why Do Multi-Agent LLM Systems Fail? In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38. Datasets and Benchmarks Track.
104. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems* **2022**.
105. Zhou, Y.; Zhang, L.; Wu, Y.; Wang, M.; Peng, B.; Liu, J.; Fan, X.; Zhao, Z. OmniOPD: Logit-Free On-Policy Distillation via Speculative Verification. *arXiv preprint arXiv:2606.01476* **2026**.
106. Lee, J.; Lu, X.; Hessel, J.; Brahman, F.; Yu, Y.; Bisk, Y.; Choi, Y.; Gabriel, S. How to Train Your Fact Verifier: Knowledge Transfer with Multimodal Open Models. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024. Association for Computational Linguistics, 2024, pp. 13060–13077.
107. Stechly, K.; Valmeekam, K.; Kambhampati, S. On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks. In Proceedings of the International Conference on Learning Representations, 2025.
108. Sun, Z.; Shen, S.; Cao, S.; Liu, H.; Li, C.; Shen, Y.; Gan, C.; Gui, L.; Wang, Y.X.; Yang, Y.; et al. Aligning Large Multimodal Models with Factually Augmented RLHF. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024. Association for Computational Linguistics, 2024, pp. 13088–13110. https://doi.org/10.18653/v1/2024.findings-acl.775.
109. Chen, X.; Shi, B.; Le, C.; Zhang, J.; Wang, K.; Gong, R.; Zhang, J.; Wang, C. Iterative Multimodal Retrieval-Augmented Generation for Medical Question Answering. *arXiv preprint arXiv:2604.27724* **2026**.
110. Wang, J.; Zhao, Z.; Lei, S.; Su, H.; Huang, Y.; Xie, Y.; Tang, K.; Xu, G.; Ye, A.; Ma, Y.; et al. DriveStack-VLA: Render-Teacher Alignment for BEV-Based DeepStack Vision-Language-Action Model. *arXiv preprint arXiv:2606.24051* **2026**.
111. Hu, Z.; Rostami, M.; Thomason, J. Multimodal Synthetic Data Finetuning and Model Collapse: Insights from VLMs and Diffusion Models. In Proceedings of the Proceedings of the 27th ACM International Conference on Multimodal Interaction (ICMI). Association for Computing Machinery, 2025.
112. Nian, Y.; Cao, H.; Zhu, S.; Zou, H.P.; Luan, Q.; Zhao, Y. When Only the Final Text Survives: Implicit Execution Tracing for Multi-Agent Attribution. *arXiv preprint arXiv:2603.17445* **2026**.
113. Zhang, S.; Yin, M.; Zhang, J.; Liu, J.; Han, Z.; Zhang, J.; Li, B.; Wang, C.; Wang, H.; Chen, Y.; et al. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. In Proceedings of the Proceedings of the 42nd International Conference on Machine Learning. PMLR, 2025, Vol. 267, *Proceedings of Machine Learning Research*.
114. Yagubyan, A. How Consistent Are LLM Agents? Measuring Behavioral Reproducibility in Multi-Step Tool-Calling Pipelines. *arXiv preprint arXiv:2605.28840* **2026**.
115. Wang, Y.; Fu, Z.; Cai, J.; Tang, P.; Lyu, H.; Fang, Y.; Zheng, Z.; Zhou, J.; Zeng, G.; Xiao, C.; et al. Ultra-FineWeb: Efficient Data Filtering and Verification for High-Quality LLM Training Data. *arXiv preprint arXiv:2505.05427* **2025**.
116. Wang, J.; Xiang, D.; Xu, J.; Liu, Z.; Zhang, Z.; Gong, G.; Fang, J.; Liu, C.; Liu, P.; Liu, T.; et al. BLADE: Scalable Bi-level Adaptive Data Selection for LLM Training, 2026, [arXiv:cs.LG/2606.18650].
117. Wang, S.; Ouyang, X.; Xu, T.; Hu, Y.; Liu, J.; Chen, G.; Zhang, T.; Zheng, J.; Yang, K.; Ren, X.; et al. OPUS: Towards Efficient and Principled Data Selection in Large Language Model Pre-training in Every Iteration, 2026, [arXiv:cs.CL/2602.05400].
118. Zhou, F.; Wang, Z.; Liu, Q.; Li, J.; Liu, P. Programming Every Example: Lifting Pre-training Data Quality Like Experts at Scale. In Proceedings of the Proceedings of the 42nd International Conference on Machine Learning. PMLR, 2025, Vol. 267, *Proceedings of Machine Learning Research*, pp. 79281–79327, [arXiv:cs.CL/2409.17115].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

119. Liu, X.; Liang, J.; Ye, M.; Xi, Z. Robustifying Safety-Aligned Large Language Models through Clean Data Curation, 2024, [arXiv:cs.CR/2405.19358].
120. Bi, B.; Liu, S.; Ren, X.; Liu, D.; Lin, J.; Wang, Y.; Mei, L.; Fang, J.; Guo, J.; Cheng, X. RefineX: Learning to Refine Pre-training Data at Scale from Expert-Guided Programs, 2025, [2507.03253].
121. Yu, Z.; Xiong, C. RePro: Training Language Models to Faithfully Recycle the Web for Pretraining, 2025, [arXiv:cs.CL/2510.10681].
122. Akter, S.N.; Prabhumoye, S.; Kamalu, J.; Satheesh, S.; Nyberg, E.; Patwary, M.; Shoeybi, M.; Catanzaro, B. MIND: Math Informed syNthetic Dialogues for Pretraining LLMs. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.AI/2410.12881].
123. Maini, P.; Dorna, V.; Doshi, P.; Carranza, A.; Pan, F.; Urbanek, J.; Burstein, P.; Fang, A.; Deng, A.; Abbas, A.; et al. BeyondWeb: Lessons from Scaling Synthetic Data for Trillion-scale Pretraining, 2025, [2508.10975].
124. Zhou, J.; Wang, Y.; Wu, X.; Yu, T.; Zhang, F. WRAP++: Web discoveRy Amplified Pretraining, 2026, [arXiv:cs.CL/2604.06829].
125. Tan, E.X.; Lanchantin, J.; Dhuliawala, S.; Li, D.; Nguyen, T.; Xu, J.; Yu, P.; Kulikov, I.; Sukhbaatar, S.; Weston, J.; et al. Self-Improving Pretraining: using post-trained models to pretrain better models, 2026, [arXiv:cs.CL/2601.21343].
126. Ruan, Y.; Band, N.; Maddison, C.J.; Hashimoto, T. Reasoning to Learn from Latent Thoughts, 2025, [2503.18866].
127. Yu, Z.; Xiong, C. Generating Pretraining Tokens from Organic Data for Data-Bound Scaling, 2026, [arXiv:cs.CL/2605.17849].
128. Fan, S.; Pagliardini, M.; Jaggi, M. DOGE: Domain Reweighting with Generalization Estimation. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning, 2024, pp. 12895–12915, [arXiv:cs.LG/2310.15393].
129. Albalak, A.; Pan, L.; Raffel, C.; Wang, W.Y. Efficient Online Data Mixing For Language Model Pre-Training, 2023, [arXiv:cs.CL/2312.02406].
130. Zhao, K.; Miao, Z.; Aizawa, A.; Tsuruoka, Y. RegMix-D: Dynamic Data Mixing via Proxy Training Trajectories, 2026, [arXiv:cs.CL/2606.18663].
131. Liu, W.; Zeng, W.; He, K.; Jiang, Y.; He, J. What Makes Good Data for Alignment? A Comprehensive Study of Automatic Data Selection in Instruction Tuning. In Proceedings of the International Conference on Learning Representations, 2024, [arXiv:cs.CL/2312.15685].
132. Wu, C.; Mao, J.; Miao, Y.; Lian, S.; Yu, B.; Lin, X.; Huang, C.; Zhang, L.; Chen, K. ScalSelect: Scalable Training-Free Multimodal Data Selection for Efficient Visual Instruction Tuning, 2026, [arXiv:cs.CV/2602.11636].
133. Huang, H.; Liu, J.; Yu, Z.; Cai, L.; Jiao, D.; Zhang, W.; Tang, S.; Li, J.; Jiang, H.; Li, H.; et al. Align2LLaVA: Cascaded Human and Large Language Model Preference Alignment for Multi-modal Instruction Curation. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2025. Association for Computational Linguistics, 2025, p. 8759–8781. https://doi.org/10.18653/v1/2025.findings-acl.458.
134. Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N.A.; Khashabi, D.; Hajishirzi, H. Self-Instruct: Aligning Language Models with Self-Generated Instructions. *Proceedings of ACL* **2023**.
135. Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; Lin, Q.; Jiang, D. WizardLM: Empowering Large Pre-Trained Language Models to Follow Complex Instructions. In Proceedings of the International Conference on Learning Representations, 2024.
136. Panagopoulou, A.; Xue, L.; Yu, N.; Li, J.; Li, D.; Joty, S.; Xu, R.; Savarese, S.; Xiong, C.; Niebles, J.C. X-InstructBLIP: A Framework for Aligning Image, 3D, Audio, Video to LLMs and its Emergent Cross-modal Reasoning. In Proceedings of the European Conference on Computer Vision (ECCV). Springer, 2024, pp. 177–197, [2311.18799]. https://doi.org/10.1007/978-3-031-72995-9_11.
137. Lin, B.; Ye, Y.; Zhu, B.; Cui, J.; Ning, M.; Jin, P.; Yuan, L. Video-LLaVA: Learning United Visual Representation by Alignment Before Projection. In Proceedings of the Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2024, pp. 5971–5984, [2311.10122]. https://doi.org/10.18653/v1/2024.emnlp-main.342.
138. Zhou, G.; Yin, Y.; Chai, W.; Tong, S.; Fu, X.; Liu, Z. VisionFoundry: Teaching VLMs Visual Perception with Synthetic Images, 2026, [arXiv:cs.CV/2604.09531].
139. Liang, Y.; Zhou, H.; Li, M.; Li, L.; Hsieh, C.J.; Zhou, T. Self-Evolving Visual Questioner, 2026, [arXiv:cs.CV/2606.13929].
140. Yuan, Y.; Li, W.; Liu, J.; Tang, D.; Luo, X.; Qin, C.; Zhang, L.; Zhu, J. Osprey: Pixel Understanding with Visual Instruction Tuning. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

and Pattern Recognition (CVPR), 2024, pp. 28202–28211, [2312.10032]. https://doi.org/10.1109/CVPR52733.2024.02664.

141. Lin, W.; Wei, X.; An, R.; Gao, P.; Zou, B.; Luo, Y.; Huang, S.; Zhang, S.; Li, H. Draw-and-Understand: Leveraging Visual Prompts to Enable MLLMs to Comprehend What You Want. In Proceedings of the International Conference on Learning Representations, 2025, [2403.20271].
142. Zhang, Y.; Zhang, R.; Gu, J.; Zhou, Y.; Lipka, N.; Yang, D.; Sun, T. LLaVAR: Enhanced Visual Instruction Tuning for Text-Rich Image Understanding, 2023, [2306.17107].
143. Kondic, J.; Li, P.; Joshi, D.; He, Z.; Abedin, S.; Sun, J.; Wiesel, B.; Schwartz, E.; Nassar, A.; Wu, B.; et al. ChartGen: Scaling Chart Understanding Via Code-Guided Synthetic Chart Generation, 2025, [2507.19492].
144. Yuan, Y.; Zhang, H.; Li, W.; Cheng, Z.; Zhang, B.; Li, L.; Li, X.; Zhao, D.; Zhang, W.; Zhuang, Y.; et al. VideoRefer Suite: Advancing Spatial-Temporal Object Understanding with Video LLM. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025, pp. 18970–18980, [2501.00599].
145. Ghazanfari, S.; Croce, F.; Flammarion, N.; Krishnamurthy, P.; Khorrami, F.; Garg, S. Chain-of-Frames: Advancing Video Understanding in Multimodal LLMs via Frame-Aware Reasoning. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2026, pp. 2746–2755, [2506.00318].
146. Li, H.; Zheng, K.; Wu, J.; Xu, C.; Sun, Q.; Hu, H.; Yang, Y. VeriEvol: Scaling Multimodal Mathematical Reasoning via Verifiable Evol-Instruct, 2026, [arXiv:cs.AI/2606.23543].
147. Yang, X.; Liu, C.; Wei, Y. Data Selection Matters: Towards Robust Instruction Tuning of Large Multimodal Models. In Proceedings of the Advances in Neural Information Processing Systems, 2025.
148. Stan, G.B.M.; Aflalo, E.; Madasu, A.; Lal, V.; Howard, P. Learning from Reasoning Failures via Synthetic Data Generation. In Proceedings of the Proceedings of the Fortieth AAAI Conference on Artificial Intelligence, 2026.
149. Liu, F.; Lin, K.; Li, L.; Wang, J.; Yacoob, Y.; Wang, L. Mitigating Hallucination in Large Multi-Modal Models via Robust Instruction Tuning. In Proceedings of the International Conference on Learning Representations, 2024, [2306.14565].
150. Kang, L.; Wang, Z.; Zhang, Y.; Wu, D.; Wang, J.; Ma, M.; Yan, H.; Wang, Z. Learning with Challenges: Adaptive Difficulty-Aware Data Generation for Mobile GUI Agent Training, 2026, [arXiv:cs.AI/2601.22781].
151. Wang, Z.; Zeng, Y.; Gong, Z.; Guo, Y.; Zhu, F.; Zhang, H.; Zhang, W.; Zuo, W. AnE: Pushing the Reasoning Frontier of Multimodal LLMs via Anchor Evolution, 2026, [arXiv:cs.CV/2605.25571].
152. Jiao, Z.; Zhang, Z.; Wang, S.; Wang, W.; Zhao, B.; Wei, H.; Zhang, L. Socratic-Geo: Synthetic Data Generation and Cross-Modal Geometric Reasoning via Multi-Agent Interaction. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2026, pp. 23795–23804.
153. Yu, C.; Xu, Y.; Chen, Y.; Zhang, W. Optimizing LVLMs with On-Policy Data for Effective Hallucination Mitigation. In Proceedings of the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2026.
154. Hung, C.Y.; Majumder, N.; Kong, Z.; Mehrish, A.; Zadeh, A.; Li, C.; Valle, R.; Catanzaro, B.; Poria, S. TangoFlux: Super Fast and Faithful Text to Audio Generation with Flow Matching and Clap-Ranked Preference Optimization. In Proceedings of the International Conference on Learning Representations, 2026.
155. Ye, Z.; Agarwal, R.; Liu, T.; Joshi, R.; Velury, S.; Le, Q.V.; Tan, Q.; Liu, Y. Reward-Guided Prompt Evolving in Reinforcement Learning for LLMs. In Proceedings of the Proceedings of the 42nd International Conference on Machine Learning, 2025, Vol. 267, *Proceedings of Machine Learning Research*, pp. 71910–71937.
156. Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* **2024**.
157. Ornith Team. Ornith-1.5: From Self-Scaffolding to Self-Improvement. Ornith Blog, 2026.
158. Xu, J.; Huang, Y.; Cheng, J.; Yang, Y.; Xu, J.; Wang, Y.; Duan, W.; Yang, S.; Jin, Q.; Li, S.; et al. VisionReward: Fine-Grained Multi-Dimensional Human Preference Learning for Image and Video Generation. *Proceedings of the AAAI Conference on Artificial Intelligence* **2026**, *40*, 11269–11277. https://doi.org/10.1609/aaai.v40i13.38107.
159. Gu, Y.; Dong, L.; Wei, F.; Huang, M. MiniLLM: Knowledge Distillation of Large Language Models. In Proceedings of the International Conference on Learning Representations, 2024.
160. Agarwal, R.; Vieillard, N.; Zhou, Y.; Stanczyk, P.; Ramos, S.; Geist, M.; Bachem, O. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In Proceedings of the International Conference on Learning Representations, 2024.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

161. Kang, W.; Galim, K.; Oh, S.; Kang, M.; Park, S.; Kim, D.; Lee, M.; Kim, M.; Tiwari, R.; Zeng, Y.; et al. AsyncOPD: How Stale Can On-Policy Distillation Be? *arXiv preprint arXiv:2606.24143* **2026**.
162. Xing, X.; Wang, H.; Gao, B.; Li, Z.; Tang, Y. Trust Region On-Policy Distillation. *arXiv preprint arXiv:2606.01249* **2026**.
163. Wang, S.; Huang, W.; Yu, X.; Yang, Z.; Lin, H.; Wu, K.; Xiao, C.; Chen, C.; Wang, W.; Zhu, B.; et al. Beyond SFT-to-RL: Pre-alignment via Black-Box On-Policy Distillation for Multimodal RL. *arXiv preprint arXiv:2604.28123* **2026**.
164. Ming, R.; Wu, H.; Hu, S.; He, Z.; Yu, B. One-Token Rollout: Guiding Supervised Fine-Tuning of LLMs with Policy Gradient. *arXiv preprint arXiv:2509.26313* **2025**.
165. Zhao, T.; Hu, Y.; Li, R.; Lu, Y.; Shi, H.; Zhu, Y.; Dou, Z. OPOD: On-Policy Omni Distillation. *arXiv preprint arXiv:2607.20918* **2026**.
166. Ye, T.; Dong, L.; Wu, X.; Huang, S.; Wei, F. On-Policy Context Distillation for Language Models. *arXiv preprint arXiv:2602.12275* **2026**.
167. Wang, Y.; Wang, Z.; Zeng, B.; Zhang, R.; Liu, W.; Yang, L.; Dai, Y.; Shi, Y.; Li, B.; Tong, C.; et al. Flux-OPD: On-Policy Distillation with Evolving Contexts. *arXiv preprint arXiv:2607.28022* **2026**.
168. Packer, C.; Wooders, S.; Lin, K.; Fang, V.; Patil, S.G.; Stoica, I.; Gonzalez, J.E. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560* **2023**.
169. Chhikara, P.; Khant, D.; Aryan, S.; Singh, T.; Yadav, D. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. In Proceedings of the ECAI 2025 - 28th European Conference on Artificial Intelligence, 2025, pp. 2993–3000. https://doi.org/10.3233/FAIA251160.
170. He, J.; Li, X.; Lin, C.H.; Yang, M.H. Reasmory: 3D Reconstruction as Explicit Memory for VLMs Spatial Reasoning. *arXiv preprint arXiv:2606.00963* **2026**.
171. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language Agents with Verbal Reinforcement Learning. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.
172. Feng, E.; Zhou, W.; Liu, Z.; Chen, L.; Dong, Y.; Zhang, C.; Zhao, Y.; Du, D.; Hua, Z.; Xia, Y.; et al. Get Experience from Practice: LLM Agents with Record & Replay. *arXiv preprint arXiv:2505.17716* **2025**.
173. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research* **2024**.
174. Lin, H.; Li, P.; Song, J.; Jiang, F.; Zhang, T. MUSE-Autoskill: Self-Evolving Agents via Skill Creation, Memory, Management, and Evaluation. *arXiv preprint arXiv:2605.27366* **2026**.
175. Yang, S.; Wu, J.; Wong, D.F.; Wang, D. SelfMem: Self-Optimizing Memory for AI Agents. *arXiv preprint arXiv:2607.03726* **2026**.
176. Liu, J.; Ye, X.; Xia, P.; Zheng, Z.; Xie, C.; Ding, M.; Yao, H. EvolveMem: Self-Evolving Memory Architecture via AutoResearch for LLM Agents. *arXiv preprint arXiv:2605.13941* **2026**.
177. Zelikman, E.; Wu, Y.; Mu, J.; Goodman, N.D. STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning. In Proceedings of the Advances in Neural Information Processing Systems, 2022, [2203.14465].
178. Gulcehre, C.; et al. Reinforced Self-Training (ReST) for Language Modeling. *arXiv preprint arXiv:2308.08998* **2023**.
179. Pan, J.; Wang, X.; Neubig, G.; Jaitly, N.; Ji, H.; Suhr, A.; Zhang, Y. Training Software Engineering Agents and Verifiers with SWE-Gym. In Proceedings of the Proceedings of the 42nd International Conference on Machine Learning, 2025.
180. Yuan, S.; Chen, Z.; Xi, Z.; Ye, J.; Du, Z.; Chen, J. Agent-R: Training Language Model Agents to Reflect via Iterative Self-Training. *arXiv preprint arXiv:2501.11425* **2025**.
181. Zhang, K.; Li, B.; Zhang, P.; Pu, F.; Cahyono, J.A.; Hu, K.; Liu, S.; Zhang, Y.; Yang, J.; Li, C.; et al. LMMs-Eval: Reality Check on the Evaluation of Large Multimodal Models. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025, 2025, pp. 881–916. https://doi.org/10.18653/v1/2025.findings-naacl.51.
182. Rahmanzadehgervi, P.; Bolton, L.; Taesiri, M.R.; Nguyen, A.T. Vision language models are blind: Failing to translate detailed visual features into words. In Proceedings of the Asian Conference on Computer Vision (ACCV), 2024, [arXiv:cs.AI/2407.06581].
183. Liu, Y.; He, J.; Su, H.; Lian, R.; Nian, Y.; Vincent, J.W.; Vishnubhotla, S.; Piramuthu, R.; Mansour, S. MDSEval: A Meta-Evaluation Benchmark for Multimodal Dialogue Summarization. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2025, 2025, pp. 14707–14727. https://doi.org/10.18653/v1/2025.findings-emnlp.794.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

184. Khan, M.S.U.R.; Suryanarayanan, S.; Anand, T.; Khapra, M.M. Seeing Isn't Believing: Uncovering Blind Spots in Evaluator Vision-Language Models. *arXiv preprint arXiv:2604.21523* **2026**.
185. Hu, Y.; Askari-Hemmat, R.; Hall, M.; Dinan, E.; Zettlemoyer, L.; Ghazvininejad, M. Multimodal RewardBench 2: Evaluating Omni Reward Models for Interleaved Text and Image. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2026, pp. 36904–36915.
186. Khanmohammadi, R.; Miahi, E.; Kaur, S.; Smiley, C.H.; Brugere, I.; Thind, K.; Ghassemi, M.M. Grounded or Guessing? LVLM Confidence Estimation via Blind-Image Contrastive Ranking. *arXiv preprint arXiv:2605.10893* **2026**.
187. Xu, R.; Wang, Z.; Fan, R.Z.; Liu, P. Benchmarking Benchmark Leakage in Large Language Models. *arXiv preprint arXiv:2404.18824* **2024**.
188. Fu, Y.V.; Uzuner, Ö.; Yetişgen, M.; Xia, F. Does Data Contamination Detection Work (Well) for LLMs? A Survey and Evaluation on Detection Assumptions. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025. Association for Computational Linguistics, 2025, pp. 5235–5256.
189. Panickssery, A.; Bowman, S.R.; Feng, S. LLM Evaluators Recognize and Favor Their Own Generations. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
190. Jain, N.; Han, K.; Gu, A.; Li, W.D.; Yan, F.; Zhang, T.; Wang, S.; Solar-Lezama, A.; Sen, K.; Stoica, I. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. In Proceedings of the International Conference on Learning Representations, 2025.
191. Xia, P.; Zeng, K.; Liu, J.; Qin, C.; Wu, F.; Zhou, Y.; Xiong, C.; Yao, H. Agent0: Unleashing Self-Evolving Agents from Zero Data via Tool-Integrated Reasoning. *arXiv preprint arXiv:2511.16043* **2025**.
192. Li, S.S.; Xin, R.; Wang, Y.; Xiao, T.; Shao, R.; Hao, Z.; Sclar, M.; Brahman, F.; Koh, P.W.; Tsvetkov, Y. EvoLM: Self-Evolving Language Models through Co-Evolved Discriminative Rubrics. In Proceedings of the International Conference on Learning Representations, 2026.
193. Xu, W.; Zhu, G.; Zhao, X.; Pan, L.; Li, L.; Wang, W. Pride and Prejudice: LLM Amplifies Self-Bias in Self-Refinement. In Proceedings of the Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2024, pp. 15474–15492. https://doi.org/10.18653/v1/2024.acl-long.826.
194. Liu, A.; Meng, J. Self-Correction as Feedback Control: Error Dynamics, Stability Thresholds, and Prompt Interventions in LLMs. *arXiv preprint arXiv:2604.22273* **2026**.
195. Zhang, J.; Hu, S.; Lu, C.; Lange, R.T.; Clune, J. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. In Proceedings of the International Conference on Learning Representations, 2026.
196. Sun, Y.; Liang, Y.; Zhang, Z.; Liu, X.; Teng, J. Theoretical Modeling of Large Language Model Self-Improvement Training Dynamics Through Solver-Verifier Gap. In Proceedings of the International Conference on Learning Representations, 2026.
197. Liu, C.; Dong, Y.; Shen, Y.; Lei, Q. A Task-Centric Theory for Iterative Self-Improvement with Easy-to-Hard Curricula. In Proceedings of the International Conference on Learning Representations, 2026.
198. Shumailov, I.; Shumaylov, Z.; Zhao, Y.; Papernot, N.; Anderson, R.J.; Gal, Y. AI models collapse when trained on recursively generated data. *Nature* **2024**, *631*, 755–759. https://doi.org/10.1038/s41586-024-07566-y.
199. Wu, T.; Li, X.; Liu, P. Progress or Regress? Self-Improvement Reversal in Post-training. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.CL/2407.05013].
200. Dohmatob, E.; Feng, Y.; Subramonian, A.; Kempe, J. Strong Model Collapse. In Proceedings of the International Conference on Learning Representations, 2025.
201. Yang, T.; Shi, Y.; Sun, R.; Huang, J.; Liu, N.; Sun, J. TRON: Targeted Rule-Verifiable Online Environments for Visual Reasoning RL. *arXiv preprint arXiv:2606.01599* **2026**.
202. Zhu, T.; Zhang, S.; Huang, J.Y.; Song, S.; Wen, X.; Li, Y.; Poon, H.; Chen, M. Video Models Can Reason with Verifiable Rewards. *arXiv preprint arXiv:2605.15458* **2026**.
203. Zheng, L.; Chiang, W.L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.P.; et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems* **2023**.
204. Jain, S.; Hamidieh, K.; Georgiev, K.; Ilyas, A.; Ghassemi, M.; Madry, A. Data Debiasing with Datamodels (D3M): Improving Subgroup Robustness via Data Selection. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
205. Kaptein, M.; Khan, V.J.; Podstavnychy, A. Runtime Governance for AI Agents: Policies on Paths. *arXiv preprint arXiv:2603.16586* **2026**.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

206. Chen, L.; Li, J.; Dong, X.; Zhang, P.; Zang, Y.; Chen, Z.; Duan, H.; Wang, J.; Qiao, Y.; Lin, D.; et al. Are We on the Right Way for Evaluating Large Vision-Language Models? In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
207. Fang, A.; Jose, A.M.; Jain, A.; Schmidt, L.; Toshev, A.; Shankar, V. Data Filtering Networks. In Proceedings of the International Conference on Learning Representations, 2024.
208. Xu, H.; Xie, S.; Tan, X.E.; Huang, P.Y.; Howes, R.; Sharma, V.; Li, S.W.; Ghosh, G.; Zettlemoyer, L.; Feichtenhofer, C. Demystifying CLIP Data. In Proceedings of the International Conference on Learning Representations, 2024.
209. Wang, S.; Long, Z.; Fan, Z.; Wei, Z.; Huang, X. Benchmark Self-Evolving: A Multi-Agent Framework for Dynamic LLM Evaluation. In Proceedings of the Proceedings of the 31st International Conference on Computational Linguistics. Association for Computational Linguistics, 2025.
210. Yasunaga, M.; Zettlemoyer, L.; Ghazvininejad, M. Multimodal RewardBench: Holistic Evaluation of Reward Models for Vision Language Models. *arXiv preprint arXiv:2502.14191* **2025**.
211. Zhang, Z.; Huang, X.; Xu, J.; Luo, Z.; Wang, X.; Wei, J.; Chen, X. VideoRewardBench: Comprehensive Evaluation of Multimodal Reward Models for Video Understanding, 2025, [arXiv:cs.CV/2509.00484].
212. Ma, Y.; Xia, H.; Gao, H.; Chen, W.; Ye, Y.; Yang, Y.; Chang, S.; Ding, M.; Li, Y.; Yuan, R.; et al. CMI-RewardBench: Evaluating Music Reward Models with Compositional Multimodal Instruction, 2026, [arXiv:cs.SD/2603.00610].
213. Deng, S.; Wang, K.; Yang, T.; Singh, H.; Tian, Y. Self-Improvement in Multimodal Large Language Models: A Survey. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2025. Association for Computational Linguistics, 2025, pp. 1987–2006.
214. Gao, L.; Schulman, J.; Hilton, J. Scaling Laws for Reward Model Overoptimization. In Proceedings of the Proceedings of the 40th International Conference on Machine Learning, 2023, pp. 10835–10866.
215. Sehgal, A.; Yuan, P.; Hu, Z.; Yue, Y.; Sun, J.J.; Chaudhuri, S. Self-Evolving Visual Concept Library using Vision-Language Critics. *arXiv preprint arXiv:2504.00185* **2025**. CVPR 2025.
216. Gu, Y.; Dong, L.; Wang, H.; Hao, Y.; Dong, Q.; Wei, F.; Huang, M. Data Selection via Optimal Control for Language Models. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.CL/2410.07064].
217. Zhao, E.; Wu, W.; Zhang, Y.; Zhao, X.; He, D. Ouroboros-Spatial: Closing the Data-Model Loop for Spatial Reasoning, 2026, [arXiv:cs.CV/2606.11719].
218. Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon, C.; et al. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073* **2022**.
219. Recchia, G.; Mangat, C.S.; Nyachhyon, J.; Sharma, M.; Canavan, C.; Epstein-Gross, D.; Abdulbari, M. Confirmation bias: A challenge for scalable oversight. *arXiv preprint arXiv:2507.19486* **2025**.
220. Wettig, A.; et al. QuRating: Selecting High-Quality Data for Training Language Models. In Proceedings of the Proceedings of the International Conference on Machine Learning, 2024, [2402.09739].
221. Zhuang, X.; Peng, J.; Ma, R.; Wang, Y.; Bai, T.; Wei, X.; Qiu, J.; Zhang, C.; Qian, Y.; He, C. Meta-rater: A Multi-dimensional Data Selection Method for Pre-training Language Models. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vienna, Austria, 2025; pp. 10856–10896, [arXiv:cs.CL/2504.14194]. https://doi.org/10.18653/v1/2025.acl-long.533.
222. Mizrahi, D.; Larsen, A.B.L.; Allardice, J.; Petryk, S.; Gorokhov, Y.; Li, J.; Fang, A.; Gardner, J.; Gunter, T.; Dehghan, A. Language Models Improve When Pretraining Data Matches Target Tasks, 2025, [arXiv:cs.CL/2507.12466].
223. Ye, J.; Feldman, V.; Talwar, K. Cram Less to Fit More: Training Data Pruning Improves Memorization of Facts, 2026, [2604.08519].
224. Turki, Y.; Sabolčec, V.; Messmer, B.; Jaggi, M. Toward Cross-Lingual Quality Classifiers for Multilingual Pretraining Data Selection, 2026, [2604.20549].
225. Farina, M.; Udandarao, V.; Nguyen, T.; Kuzucu, S.; Böther, M.; Hochlehnert, A.; Ghosh, A.; Nezhurina, M.; Roth, K.; Struber, J.; et al. DataComp-VLM: Improved Open Datasets for Vision-Language Models, 2026, [arXiv:cs.CV/2606.28551].
226. Yang, Z.; Huang, L.; Liu, H. Layer-Aware Influence for Online Data Valuation Estimation, 2025, [arXiv:cs.LG/2510.16007].

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

227. Nguyen, T.; Li, Y.; Golovneva, O.; Zettlemoyer, L.; Oh, S.; Schmidt, L.; Li, X. Recycling the Web: A Method to Enhance Pre-training Data Quality and Quantity for Language Models. In Proceedings of the Conference on Language Modeling (COLM), 2025, [2506.04689].
228. Fujii, K.; Tajima, Y.; Mizuki, S.; Kawamura, M.; Shimada, H.; Shiotani, T.; Saito, K.; Oi, M.; Nakamura, T.; Okamoto, T.; et al. Rewriting Pre-Training Data Boosts LLM Performance in Math and Code. In Proceedings of the International Conference on Learning Representations, 2026, [arXiv:cs.LG/2505.02881].
229. Huang, H.; Huo, Y.; Zhao, Z.; Lu, H.; Wu, S.; Wang, B.; Liu, Q.; Chen, W.; Wang, L. Beyond Filtering: Adaptive Image-Text Quality Enhancement for MLLM Pretraining, 2024, [arXiv:cs.CV/2410.16166].
230. Zhu, G.; Liu, Z.; Hou, Z.; Wang, P.; Sang, Z.; Yu, Y.; Ni, M.; Wang, W.; Gu, Y.; Cai, S.; et al. Beyond Captions: Context-Grounded Reconstruction for Biomedical Multimodal Continued Pretraining, 2026, [arXiv:cs.CL/2606.01049].
231. Niklaus, J.; Yamaguchi, A.; Štefánik, M.; Penedo, G.; Kydlíček, H.; Bakouch, E.; Tunstall, L.; Beeching, E.E.; Frere, T.; Raffel, C.; et al. How Can We Synthesize High-Quality Pretraining Data? A Systematic Study of Prompt Design, Generator Model, and Source Data, 2026, [2604.13977].
232. Arannil, V.; Narwal, N.; Bhabesh, S.S.; Thirandas, S.N.; Wang, D.Y.B.; Horwood, G.; Chirayath, A.A.; Pandeshwar, G. DoPAMine: Domain-specific Pre-training Adaptation from seed-guided data Mining, 2024, [arXiv:cs.CL/2410.00260].
233. Zeng, A.; Du, Z.; Liu, M.; Zhang, L.; Jiang, S.; Dong, Y.; Tang, J. Scaling Speech-Text Pre-training with Synthetic Interleaved Data. In Proceedings of the International Conference on Learning Representations, 2025, [2411.17607].
234. Hui, C. Towards robust long-context understanding of large language model via active recap learning, 2026, [arXiv:cs.CL/2601.13734].
235. Liu, Q.; Zheng, X.; Muennighoff, N.; Zeng, G.; Dou, L.; Pang, T.; Jiang, J.; Lin, M. RegMix: Data Mixture as Regression for Language Model Pre-training. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.CL/2407.01492].
236. Diao, S.; Yang, Y.; Fu, Y.; Dong, X.; Su, D.; Kliegl, M.; Chen, Z.; Belcak, P.; Suhara, Y.; Yin, H.; et al. Nemotron-CLIMB: Clustering-based Iterative Data Mixture Bootstrapping for Language Model Pre-training. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38, [arXiv:cs.CL/2504.13161]. Datasets and Benchmarks Track.
237. Luo, Z.; Zhang, X.; Liu, X.; Li, H.; Gong, Y.; Chen, Q.; Cheng, P. Velocitune: A Velocity-based Dynamic Domain Reweighting Method for Continual Pre-training. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2025, pp. 16644–16656, [arXiv:cs.CL/2411.14318].
238. Chen, M.F.; Hu, M.Y.; Lourie, N.; Cho, K.; Ré, C. Aioli: A Unified Optimization Framework for Language Model Data Mixing. In Proceedings of the International Conference on Learning Representations, 2025, [arXiv:cs.LG/2411.05735].
239. Fan, S.; Grangier, D.; Ablin, P. Dynamic Gradient Alignment for Online Data Mixing, 2024, [arXiv:cs.LG/2410.02498].
240. Li, Z.; Deng, Y.; Zhong, P.; Razaviyayn, M.; Mirrokni, V. PiKE: Adaptive Data Mixing for Large-Scale Multi-Task Learning Under Low Gradient Conflicts, 2025, [arXiv:cs.LG/2502.06244].
241. Wang, Y.; Liu, B.; Liu, F.; Guo, Y.; Deng, J.; Wu, X.; Zhou, W.; Zhou, X.; Wang, T. TiKMiX: Take Data Influence into Dynamic Mixture for Language Model Pre-training, 2025, [arXiv:cs.LG/2508.17677].
242. Hu, M.Y.; Gandhi, A.; Cho, K.; Linzen, T.; Sharma, P. Always Learning, Always Mixing: Efficient and Simple Data Mixing All The Time, 2026, [arXiv:cs.CL/2605.15220].
243. Ma, J.; Dang, C.; Liao, M. AC-ODM: Actor–Critic Online Data Mixing for Sample-Efficient LLM Pretraining. In Proceedings of the Proceedings of the 43rd International Conference on Machine Learning, 2026, [arXiv:cs.LG/2505.23878].
244. Yang, K.; Liu, X.; Ji, L.; Li, H.; Liang, X.; Liu, Z.; Gong, Y.; Cheng, P.; Yang, M. Data Mixing Agent: Learning to Re-weight Domains for Continual Pre-training. In Proceedings of the Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2026, pp. 9447–9473, [arXiv:cs.LG/2507.15640]. https://doi.org/10.18653/v1/2026.acl-long.427.
245. Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C.D.; Ermon, S.; Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *Advances in Neural Information Processing Systems* **2023**.
246. Wang, F.; Zhou, W.; Huang, J.Y.; Xu, N.; Zhang, S.; Poon, H.; Chen, M. mDPO: Conditional Preference Optimization for Multimodal Large Language Models. In Proceedings of the Proceedings of the 2024 Conference

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2024, pp. 8078–8088.
247. Song, M.; Zheng, M. A Survey of On-Policy Distillation for Large Language Models. *arXiv preprint arXiv:2604.00626* **2026**.
248. Yuan, Q.; Lou, J.; Yu, X.; Lin, H.; Sun, L.; Han, X.; Lu, Y. Vision-OPD: Learning to See Fine Details for Multimodal LLMs via On-Policy Self-Distillation. *arXiv preprint arXiv:2605.18740* **2026**.
249. Liu, R.; Lv, X.; Li, G.; Zhu, X.; Wang, Z.; Zhang, Z.; Chen, J.; Li, Z.; Li, B.; Gao, J.; et al. Visual-Advantage On-Policy Distillation for Vision-Language Models. *arXiv preprint arXiv:2605.21924* **2026**.
250. Huang, X.; Li, Z.; He, G.; Zhou, M.; Shechtman, E. Self Forcing: Bridging the Train-Test Gap in Autoregressive Video Diffusion. In Proceedings of the Advances in Neural Information Processing Systems, 2025.
251. Li, Q.; Yu, J.; Jiang, K.; Wei, Y.; Xing, Z.; Li, P.; Chu, R.; Zhang, S.; Liu, Y.; Wu, Z. DiffusionOPD: A Unified Perspective of On-Policy Distillation in Diffusion Models. *arXiv preprint arXiv:2605.15055* **2026**.
252. Hu, J.; Zhu, D.; Luo, X.; Zhang, D.; He, S.; Lei, Y.; Zheng, H.; Feng, S.; He, J.; Sun, Y.; et al. CORD: Bridging the Audio-Text Reasoning Gap via Weighted On-Policy Cross-Modal Distillation. *arXiv preprint arXiv:2601.16547* **2026**.
253. Cao, D.; Fu, D.; Yu, H.; Zheng, S.; Tan, X.; Jin, T. X-OPD: Cross-Modal On-Policy Distillation for Capability Alignment in Speech LLMs. *arXiv preprint arXiv:2603.24596* **2026**.
254. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems* **2020**.
255. Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; van den Driessche, G.; Lespiau, J.B.; Damoc, B.; Clark, A.; et al. Improving Language Models by Retrieving from Trillions of Tokens. In Proceedings of the Proceedings of the 39th International Conference on Machine Learning, 2022, pp. 2206–2240.
256. Yu, S.; Tang, C.; Xu, B.; Cui, J.; Ran, J.; Yan, Y.; Liu, Z.; Wang, S.; Han, X.; Liu, Z.; et al. VisRAG: Vision-based Retrieval-augmented Generation on Multi-modality Documents. In Proceedings of the International Conference on Learning Representations, 2025.
257. Faysse, M.; Sibille, H.; Wu, T.; Omrani, B.; Viaud, G.; Hudelot, C.; Colombo, P. ColPali: Efficient Document Retrieval with Vision Language Models. In Proceedings of the International Conference on Learning Representations, 2025.
258. Asadi, M.; O'Sullivan, J.W.; Cao, F.; Nedaee, T.; Rajabalifardi, K.; Li, F.F.; Adeli, E.; Ashley, E. MIRAGE: The Illusion of Visual Understanding, 2026, [arXiv:cs.AI/2603.21687].
259. Song, E.; Chai, W.; Wang, G.; Zhang, Y.; Zhou, H.; Wu, F.; Chi, H.; Guo, X.; Ye, T.; Zhang, Y.; et al. MovieChat: From Dense Token to Sparse Memory for Long Video Understanding. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 18221–18232. https://doi.org/10.1109/CVPR52733.2024.01725.
260. He, B.; Li, H.; Jang, Y.K.; Jia, M.; Cao, X.; Shah, A.; Shrivastava, A.; Lim, S.N. MA-LMM: Memory-Augmented Large Multimodal Model for Long-Term Video Understanding. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024.
261. Fan, Y.; Ma, X.; Wu, R.; Du, Y.; Li, J.; Gao, Z.; Li, Q. VideoAgent: A Memory-augmented Multimodal Agent for Video Understanding. In Proceedings of the Computer Vision – ECCV 2024. Springer, 2024, Vol. 22, pp. 75–92.
262. Chen, J.; Ye, L.; He, J.; Wang, Z.Y.; Khashabi, D.; Yuille, A. Efficient Large Multi-modal Models via Visual Context Compression. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
263. Li, X.; Wang, Y.; Yu, J.; Zeng, X.; Zhu, Y.; Huang, H.; Gao, J.; Li, K.; He, Y.; Wang, C.; et al. VideoChat-Flash: Hierarchical Compression for Long-Context Video Modeling. In Proceedings of the International Conference on Learning Representations, 2026.
264. Chen, Y.; Tan, W.; Yu, X.; Wang, Y.; Cheng, X.; Guan, K.; Jiang, H.; Li, X.; Zhu, G.; Song, R. AVOC: Enhancing Hour-Level Audio-Video Understanding in Omni-Modal LLMs via Retrieval-Inspired Token Compression. *arXiv preprint arXiv:2606.24286* **2026**.
265. Xu, W.; Liang, Z.; Mei, K.; Gao, H.; Tan, J.; Zhang, Y. A-MEM: Agentic Memory for LLM Agents. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38.
266. Maharana, A.; Lee, D.H.; Tulyakov, S.; Bansal, M.; Barbieri, F.; Fang, Y. Evaluating Very Long-Term Conversational Memory of LLM Agents. In Proceedings of the Proceedings of the 62nd Annual Meeting of

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2024.
267. Wu, D.; Wang, H.; Yu, W.; Zhang, Y.; Chang, K.W.; Yu, D. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. In Proceedings of the International Conference on Learning Representations, 2025.
268. Wei, T.; Sachdeva, N.; Coleman, B.; He, Z.; Bei, Y.; Ning, X.; Ai, M.; Li, Y.; He, J.; Chi, E.H.; et al. Evo-Memory: Benchmarking LLM Agent Test-time Learning with Self-Evolving Memory. *arXiv preprint arXiv:2511.20857* **2025**.
269. Tavakoli, M.; Salemi, A.; Ye, C.; Abdalla, M.; Zamani, H.; Mitchell, J.R. Beyond a Million Tokens: Benchmarking and Enhancing Long-Term Memory in LLMs. In Proceedings of the International Conference on Learning Representations, 2026, [arXiv:cs.CL/2510.27246].
270. Xiong, Z.; Lin, Y.; Xie, W.; He, P.; Liu, Z.; Tang, J.; Lakkaraju, H.; Xiang, Z. How Memory Management Impacts LLM Agents: An Empirical Study of Experience-Following Behavior. In Proceedings of the Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), San Diego, California, United States, 2026; pp. 623–645. https://doi.org/10.18653/v1/2026.acl-long.27.
271. Zhao, A.; Huang, D.; Xu, Q.; Lin, M.; Liu, Y.J.; Huang, G. ExpeL: LLM Agents Are Experiential Learners. *Proceedings of the AAAI Conference on Artificial Intelligence* **2024**, *38*, 19632–19642. https://doi.org/10.1609/aaai.v38i17.29936.
272. Yan, D.; Zhang, H.; Huzhang, G.; Li, Y.; Wang, Y.; Chen, Q.G.; Xu, Z.; Luo, W.; Li, Y.; Dong, W.; et al. M^2: Dual-Memory Augmentation for Long-Horizon Web Agents via Trajectory Summarization and Insight Retrieval. *arXiv preprint arXiv:2603.00503* **2026**.
273. Wang, J.; Yan, Q.; Wang, Y.; Tian, Y.; Mishra, S.S.; Xu, Z.; Gandhi, M.; Xu, P.; Cheong, L.L. Reinforcement Learning for Self-Improving Agent with Skill Library. *arXiv preprint arXiv:2512.17102* **2025**.
274. Huang, A.; Block, A.; Foster, D.J.; Rohatgi, D.; Zhang, C.; Simchowitz, M.; Ash, J.T.; Krishnamurthy, A. Self-Improvement in Language Models: The Sharpening Mechanism. In Proceedings of the International Conference on Learning Representations, 2025.
275. Ferbach, D.; Bertrand, Q.; Bose, A.J.; Gidel, G. Self-Consuming Generative Models with Curated Data Provably Optimize Human Preferences. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
276. Norman, J.D.; Rivera, M.U.; Hughes, D.A. Reliability without Validity: A Systematic, Large-Scale Evaluation of LLM-as-a-Judge Models Across Agreement, Consistency, and Bias. *arXiv preprint arXiv:2606.19544* **2026**.
277. Yang, L.; Zhang, X.; Tian, Y.; Zhang, S.; Shang, C.; Xu, M.; Zhang, W.; Cui, B. HermesFlow: Seamlessly Closing the Gap in Multimodal Understanding and Generation. In Proceedings of the Advances in Neural Information Processing Systems, 2025, Vol. 38.
278. Tong, Y.; Chang, D.; Yin, Z.; Liu, X.; Fang, Y.; Ma, Z. Reversing the Flow: Generation-to-Understanding Synergy in Large Multimodal Models. *arXiv preprint arXiv:2605.15792* **2026**.
279. Zhang, J.; Li, T.; Li, L.; Yang, Z.; Cheng, Y. Are Unified Vision-Language Models Necessary: Generalization Across Understanding and Generation. *arXiv preprint arXiv:2505.23043* **2025**.
280. Liu, J.; Shuai, X.; Ding, H.; Jiang, Y.G. Unison: Benchmarking Unified Multimodal Models via Synergistic Understanding and Generation. *arXiv preprint arXiv:2606.26984* **2026**.
281. Lee, M.; Kahng, M. Data-Prompt Co-Evolution: Growing Test Sets to Refine LLM Behavior. *arXiv preprint arXiv:2510.12728* **2025**.
282. Luo, R.; Xia, X.; Wang, L.; Chen, L.; Shan, R.; Luo, J.; Yang, M.; Chua, T.S. NExT-OMNI: Towards Any-to-Any Omnimodal Foundation Models with Discrete Flow Matching. In Proceedings of the International Conference on Learning Representations, 2026.
283. Guo, Q.; Song, K.; Feng, Z.; Ma, Z.; Zhang, Q.; Gao, S.; Yu, X.; Sun, Y.; Chang, T.W.; Chen, J.; et al. M2-omni: Advancing Omni-MLLM for Comprehensive Modality Support with Competitive Performance. *arXiv preprint arXiv:2502.18778* **2025**.
284. Driess, D.; Xia, F.; Sajjadi, M.S.M.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; et al. PaLM-E: An Embodied Multimodal Language Model. In Proceedings of the Proceedings of the 40th International Conference on Machine Learning, 2023, pp. 8469–8488.
285. Zenil, H. On the Limits of Self-Improving in Large Language Models: The Singularity Is Not Near Without Symbolic Model Synthesis. *arXiv preprint arXiv:2601.05280* **2026**.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 September 2026 doi:10.20944/preprints202609.0154.v1

286. Goel, S.; Strüber, J.; Auzina, I.A.; Chandra, K.K.; Kumaraguru, P.; Kiela, D.; Prabhu, A.; Bethge, M.; Geiping, J. Great Models Think Alike and this Undermines AI Oversight. In Proceedings of the Proceedings of the 42nd International Conference on Machine Learning, 2025.
287. Lang, H.; Huang, F.; Li, Y. Debate Helps Weak-to-Strong Generalization. In Proceedings of the Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence, 2025.
288. Carro, M.V.; Mester, D.A.; Nieto, F.; Stanchi, O.A.; Bergman, G.E.; Leiva, M.A.; Sprejer, E.; Gangi, L.N.F.; et al. AI Debaters are More Persuasive when Arguing in Alignment with Their Own Beliefs. *arXiv preprint arXiv:2510.13912* **2025**.
289. Longpre, S.; Mahari, R.; Obeng-Marnu, N.; Brannon, W.; South, T.; Gero, K.; Pentland, S.; Kabbara, J. Data Authenticity, Consent, & Provenance for AI are all broken: what will it take to fix them? *arXiv preprint arXiv:2404.12691* **2024**.
290. Xu, M.; Yang, Z.; Wang, Y.; Wang, Y.; Liu, X.; Dou, Z.; Hong, W.; Gu, X.; Xu, B.; Tang, J. Video2Code: Generating Interactive Webpages from UI Videos via Action-Aware Revisit. *arXiv preprint arXiv:2606.20711* **2026**.
291. Wang, Z.; Xu, C.; Liu, B.; Wang, Y.; Han, S.; Yao, Z.; Yao, H.; He, Y. Agent World Model: Infinity Synthetic Environments for Agentic Reinforcement Learning. *arXiv preprint arXiv:2602.10090* **2026**.
292. Tu, D.; Hao, H.; Yang, H.; Chen, Y.; Zhang, Y.K.; Xia, Z.; Yang, Y.; Sun, Y.; Liu, X.; Shen, F.; et al. ScaleEnv: Scaling Environment Synthesis from Scratch for Generalist Interactive Tool-Use Agent Training. *arXiv preprint arXiv:2602.06820* **2026**.
293. Guo, Y.; Lee, T.; Shi, L.X.; Chen, J.; Liang, P.; Finn, C. VLAW: Iterative Co-Improvement of Vision-Language-Action Policy and World Model. *arXiv preprint arXiv:2602.12063* **2026**.
294. Zhu, K.; Jin, Z.; Huang, D.; Yuan, H.; Hao, Y.; Liu, K.; Zhao, J. Beyond Simply Environment Scaling: Designing Effective Environment Distributions for Multimodal Agent Learning. *arXiv preprint arXiv:2608.03571* **2026**.
295. Roth, A.; Samanta, A.; Halevy, M.; Levine, Y.; Efroni, Y. Hack-Verifiable Environments: Towards Evaluating Reward Hacking at Scale. *arXiv preprint arXiv:2605.20744* **2026**.
296. Chen, D.; Huang, X.; Hu, Z.; Shi, Q.; Li, D.; Zhou, T. Sandboxed Coding Agents are Competitive Omni-modal Task Solvers. *arXiv preprint arXiv:2606.00579* **2026**.
297. Grosse, R.; Bae, J.; Anil, C.; Elhage, N.; Tamkin, A.; Tajdini, A.; Steiner, B.; Li, D.; et al. Studying Large Language Model Generalization with Influence Functions. *arXiv preprint arXiv:2308.03296* **2023**.
298. Li, Z.; Zhao, W.; Li, Y.; Sun, J. Do Influence Functions Work on Large Language Models? In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2025. Association for Computational Linguistics, 2025, pp. 14367–14382.
299. Nie, D. Post-Training is About States, Not Tokens: A State Distribution View of SFT, RL, and On-Policy Distillation. *arXiv preprint arXiv:2605.22731* **2026**.
300. Chen, X.; Zhang, S.; Wu, J. f-OPD: Stabilizing Long-Horizon On-Policy Distillation with Freshness-Aware Control. *arXiv preprint arXiv:2605.17862* **2026**.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.