arXiv:2606.04261v2 [cs.AI] 19 Sep 2026

Can Generalist Agents Automate Data Curation?

Feiyang Kang[1,*], Hanze Li[1,*], Adam Nguyen[1], Mahavir Dabas[1], Jiaqi W. Ma[2], Frederic Sala[3], Dawn Song[4], Ruoxi Jia[1],

[1]Virginia Tech, [2]University of Illinois Urbana-Champaign,
[3]University of Wisconsin-Madison, [4]University of California, Berkeley

[*]Equal Contribution **Correspondence:** fyk@vt.edu

Abstract

Curating training data is among the most consequential yet labor-intensive parts of modern AI development: practitioners iteratively propose, implement, evaluate, and revise data policies against noisy benchmark feedback. We ask whether generalist coding agents can automate this data-curation loop. We introduce CURATION-BENCH, an agent-centric benchmark that fixes the model, training recipe, and evaluation suite while giving agents command-line access to inspect data, implement policies, submit them to a fixed training/evaluation pipeline, and revise. In a vision-language instruction-tuning instantiation, out-of-the-box agents reach strong published data-selection baselines within ten iterations. However, trajectory analysis reveals a persistent *execution–research gap*: agents mainly tune local policy variants rather than explore new policy families, even when given strategy guides and paper references. Scaffolds requiring each iteration to cite, instantiate, and adapt a prior method shift agents toward method-guided exploration. The scaffolded agent autonomously composes—without human design input—a data-selection policy that outperforms strong published baselines at one-tenth their data budget. Overall, current agents can run the curation loop, but reliable data research requires scaffolded method adaptation, not open-ended prompting alone.[1]

1 Introduction

Training data is now a first-class design variable in modern AI: it shapes what models can do, how they fail, and how efficiently they learn. Prior work has automated many individual data operations, including pruning (Sorscher et al., 2022; Paul et al., 2021), deduplication (Lee et al., 2022; Abbas et al., 2023; Kang et al., 2025b), remixing (Xie et al., 2023a; Fan et al., 2024; Chen et al., 2026), rewriting (Wang et al., 2024b; Maini et al., 2024), and synthetic generation (Wang et al., 2023; Taori et al., 2023; Kang et al., 2025a). What remains largely manual is deciding which technique to use, how to adapt it, and when to revise it after noisy benchmark feedback. Because no technique is universally effective (Zhang et al., 2025; Goyal et al., 2024), practitioners must repeatedly propose, implement, evaluate, and refine data policies. This iterative search is where human effort accumulates. We ask whether the search process itself can be automated.

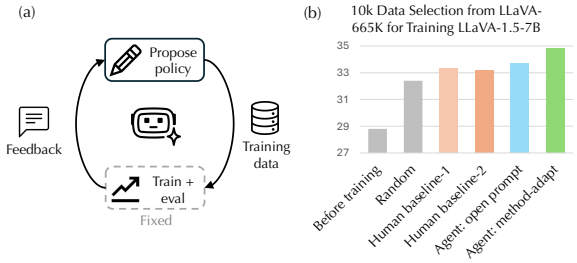


Figure 1: **Agentic data curation requires more than open-ended prompting.** (a) We formulate data curation as a policy-search loop: the agent proposes a data policy, constructs training data, observes feedback from fixed training and evaluation, and revises the policy. (b) On 10k-example selection from LLaVA-665K dataset for LLaVA-1.5-7B training, open-ended prompting improves over random selection and human-designed baselines (Wu et al., 2024; Yang et al., 2025), but method-adaptation scaffolding achieves the best result by adapting prior data-selection methods into a stronger policy.

Prior work captures two ingredients needed for automated data curation—controlled data evaluation and iterative experimentation—but not their combination. Data-centric benchmarks such as DataComp (Gadre et al., 2023; Li et al., 2024) fix the model, training recipe, and evaluation so that data becomes the variable of interest. This isolation principle is essential, but the submitted artifact is a static policy, and the benchmark does not evaluate how that policy was discovered, revised, or justified. Autonomous ML-research agents (Yang et al., 2024; Lu et al., 2024; Kon et al., 2026; Karpathy, 2026) evaluate iterative behavior, but typically optimize training recipes—code, hyperparam-

[1]Code and benchmark available at: https://github.com/feiyang-k/curation-bench .

eters, architectures, or model dimensions—under a fixed data distribution (Chan et al., 2024; Karpathy, 2026). Data-science agents (Pinchuk, 2026; Jing et al., 2024) iterate over analysis pipelines and predictive models, treating data as input rather than as the object of optimization. What remains missing is an agentic benchmark for iterative data-policy discovery.

We introduce CURATION-BENCH to fill this gap. CURATION-BENCH inherits DataComp's data-isolation principle by fixing the model, training recipe, and evaluation suite, but replaces static policy submission with an interactive terminal loop: the agent inspects the candidate pool, implements curation policies, submits them to a fixed training/evaluation pipeline, observes feedback, and revises. CURATION-BENCH records the full curation trajectory—policy scripts, data manifests, training outputs, and evaluation logs—so that agents can be evaluated not only by final accuracy, but also by how they search: whether they explore new data-policy families, ground decisions in prior evidence, or fall back on shallow local adjustments, and whether those choices improve performance.

We study the most immediate candidates for automating iterative data-policy discovery: generalist coding agents, off-the-shelf terminal systems not designed specifically for data curation. They can already inspect files, write scripts, launch jobs, parse logs, and recover from failures. We ask how far such generic execution capabilities can go in data curation, and what additional guidance helps agents make better data-policy decisions.

We instantiate CURATION-BENCH primarily in multimodal instruction tuning, where candidate pools mix heterogeneous data sources, evaluations cover diverse visual and linguistic skills, and no single selection heuristic is uniformly effective; we further include a smaller-scale CLIP-style pretraining setting to test whether the framework extends beyond instruction tuning alone. The main instantiation yields two findings. First, generalist agents are already useful data-curation executors: under open-ended prompting, they reach strong published data-selection baselines within ten iterations, recovering roughly 60% of the full-data fine-tuning gain at 1.5% of the original pool. Second, trajectory diagnostics reveal a persistent execution–research gap: agents reliably run the loop but mainly tune local policy variants—source ratios, length thresholds, and random seeds—rather than explore distinct policy families. This persists even with strategy guides and paper references, suggesting that the bottleneck is not missing methodological knowledge but failure to operationalize it into executable data policies. We therefore study scaffolds as interventions and find that mandatory citation, instantiation, and adaptation of prior methods shift agents toward method-guided exploration of new policy families. Under the strongest scaffold, the agent composes an EL2N-style top-loss policy (Paul et al., 2021) with an assistant-loss noise filter, producing a hybrid loss-based data-selection policy that outperforms the compared published non-agent baselines using one-tenth as much data.

Finally, longer-session experiments suggest that agent search iterations are a meaningful form of curation compute. When we increase the agent budget beyond ten iterations, average outcomes continue to improve up to 50 iterations without a clear plateau. This connects agentic data curation to a broader question of how to improve performance in a *finite-data, increasing-compute* regime: when additional raw data is unavailable or increasingly costly, extra compute can be spent not only on reusing data during training or expanding it synthetically (Muennighoff et al., 2023; Kim et al., 2026), but also on searching over how to select, adapt, and validate the data already available.

2 CURATION-BENCH

This section specifies the benchmark contract used throughout the paper. A CURATION-BENCH task presents an agent with a candidate data pool and a fixed training–evaluation pipeline, and asks the agent to produce a budgeted training set through an executable data policy. The benchmark constrains what the agent may change, validates each submitted dataset before training, and records the full sequence of policy attempts and outcomes. This contract separates two evaluation targets: the quality of the final curated data and the quality of the agent's search process. We first describe the design principles that define this contract, then give the task formalization, environment interface, and scoring metrics.

Design principles. The benchmark contract follows four principles. **(P1) Data isolation.** The model architecture, optimizer, training schedule, and evaluation suite are fixed by the harness. The agent's only controlled variable is the curated data. **(P2) Terminal realism.** The agent operates through a standard terminal workspace, mirroring

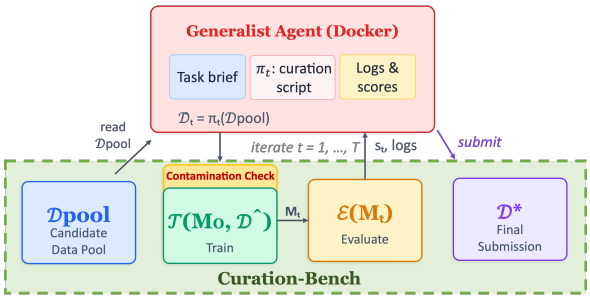


Figure 2: **Architecture of CURATION-BENCH.** A coding agent inspects the candidate pool, implements data policies, and submits curated datasets. The harness validates each submission and scores it with a fixed training–evaluation pipeline.

how current generalist coding agents interact with repositories: inspecting files, writing scripts, running commands, debugging failures, and reading logs. **(P3) Contamination control.** Every submitted dataset must pass an automatic audit before training. The audit checks exact matches and high-overlap text spans against evaluation question–answer pairs, preventing direct leakage from becoming a trivial path to improvement. **(P4) Trajectory legibility.** Each iteration persists the curation script, generated manifest, audit result, training output, evaluation scores, and run notes under a commit hash. These records support the trajectory diagnostics in Section 2.2.

2.1 Environment and Tasks

A CURATION-BENCH task is defined by $\tau = (M_0,\ \mathcal{D}_{\text{pool}},\ B,\ \mathcal{T},\ \mathcal{E},\ \mathcal{C})$, where M_0 is the initial model, $\mathcal{D}_{\text{pool}}$ is the candidate training pool, B is the data budget, $\mathcal{T}$ is the fixed training procedure, $\mathcal{E}$ is the evaluation suite, and $\mathcal{C}$ specifies constraints such as schema, budget, and contamination rules. The agent produces a data policy π that maps the pool to a curated dataset $\mathcal{D}_\pi = \pi(\mathcal{D}_{\text{pool}}),\quad |\mathcal{D}_\pi| \le B$, and the resulting policy is scored by $S(\pi) = \mathcal{E}(\mathcal{T}(M_0,\ \mathcal{D}_\pi))$.

The agent operates in a Dockerized terminal workspace containing a task brief, read-only access to the candidate pool, a starter random-selection script, a logging directory for prior iterations, and a benchmark CLI. The CLI supports dataset validation, contamination audit, training, and evaluation. The agent may edit the curation script and materialize new datasets, but may not modify the training recipe, evaluation code, or contamination checks.

At iteration t, the agent writes or revises a curation script π_t, materializes a candidate dataset $\mathcal{D}_t$, and submits it through the CLI. Submission is gated: the harness validates schema and budget, runs the contamination audit (Appendix B.2), and only then launches training. Failed submissions are returned to the agent before training. Accepted submissions are trained from M_0 using $\mathcal{T}$, evaluated by $\mathcal{E}$, and returned to the agent with aggregate and per-benchmark scores. The agent may then use this feedback to design π_{t+1}.

Table 1: Rubric for annotating each agent iteration. Labels are assigned with LLMs (Claude Opus 4.7).

Label	Criterion	Example
New policy family	Moves to a different type of data strategy.	Moves from length filtering to training-dynamics scoring.
Grounded	Justified by concrete evidence.	Uses evaluation errors, data audits, prior runs, literature, or ablations.
Effective	Improves the target metric under the same budget and training recipe.	Beats the previous policy under controlled comparison.
Shallow	Makes a reactive local adjustment with weak rationale.	"This helped, so add more," without a task-specific hypothesis.

2.2 Scoring and Trajectory Diagnostics

Agent sessions are evaluated along two axes: downstream outcome and trajectory quality.

Outcome metrics. For a session of T iterations, the primary outcome is the best score achieved during the session. We report absolute score, improvement over the base model and over random selection at the same budget, and the fraction of the full-data fine-tuning gain recovered (%) when possible. The latter metric provides a more straightforward measure of data efficacy.

Trajectory diagnostics. Outcome scores do not fully characterize the quality of agentic data curation: a high score may come from systematic exploration, but it may also come from repeated local tuning or lucky variation. We therefore annotate each iteration with four labels: *new policy family*, *grounded*, *effective*, and *shallow*, following the rubrics in Table 1. We report these labels both at the iteration level, for individual inspect–curate–train–evaluate loops, and at the trajectory level, for the full multi-iteration session. These diagnostics make the agent's search process auditable and show where the search stalls.

3 Default Agents: Useful Executors, Narrow Researchers

3.1 Open-Prompt Agent Setup

We evaluate generalist coding agents in the CURATION-BENCH environment under the mini-

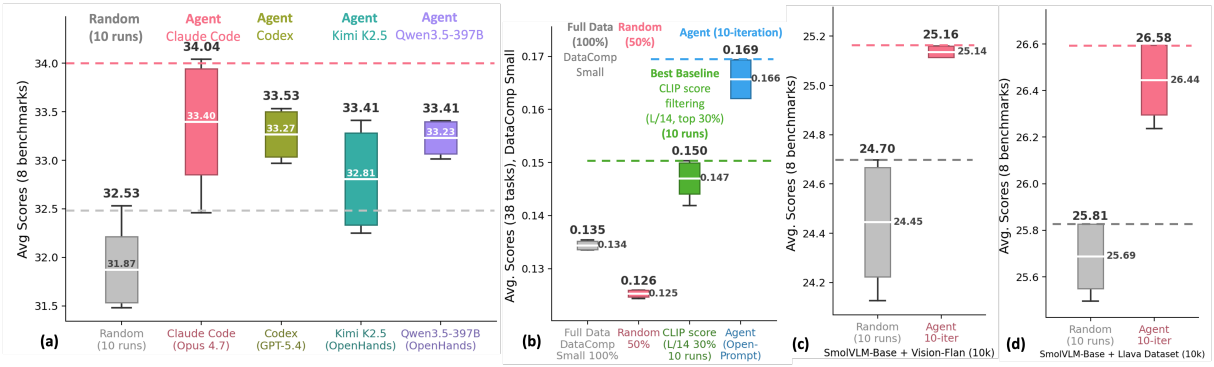


Figure 3: **Agent data-curation results (open-prompt).** The bars and the centerline show the average scores and the range of standard deviation, and the upper/lower wicks showing the min/max outcome of a session. Agents run 3 sessions each with 10 iterations, and the baseline is repeated for 10 independent runs. 100% iterations are successfully executed. **(a).** Fine-tuning pre-trained LLaVA-1.5-7B on curated subsets of 10k samples from the LLaVA-665K instruction-tuning pool and evaluate on 8 benchmarks. All agents–Claude Code (Opus 4.7), Codex (GPT-5.4), and Kimi K2.5/Qwen3.5-397B with OpenHands harness–significantly improve training outcome compared to training on random data. **(b).** Pre-training ViT-B/32 CLIP models for the DataComp Small benchmark (Filtering track) and reporting average scores from the 38 tasks. The agent (Claude Code) established a notable margin over the best baseline (top 30% CLIP L/14 scores). **(c)/(d).** Fine-tuning SmolVLM-Base on curated subsets of 10k samples from the LLaVA-665K/Vision-Flan 186k instruction-tuning datasets and evaluate on 8 benchmarks, respectively. The agent (Claude Code) substantially improves training outcome compared to training on random data.

mal scaffold with open-prompt. The agent receives only the benchmark interface: the task objective, workspace, constraints, and commands for validation, training, and evaluation. It must decide on its own what data policy to implement and how to revise that policy across iterations. It follows the standard inspect–curate–train–evaluate–log loop (Prompt text in Appendix G):

Open-prompt loop.

1. Read the task brief and constraints.
2. Inspect the candidate pool and previous runs.
3. Propose a data policy, such as selection, mixing, filtering, scoring, or rewriting.
4. Materialize a training set and save a manifest.
5. Fine-tune the model with the fixed training recipe.
6. Evaluate on the fixed benchmark suite.
7. Write a run note explaining the hypothesis, evidence, and next action.

This setup gives the agent procedural access to the curation loop while leaving the choice of data policy to the agent.

3.2 Outcome: Generalist Agents Match Human-Designed Baselines

We score each agent session using the outcome and trajectory metrics defined in Section 2.2. Experimental details, additional results, and ablation studies are provided in Appendix D.

We also report the rate of **Execution success:** whether the agent completes the required loop without violating task constraints—producing a valid curated dataset, interacting with the benchmark, and logging the result. We find that execution is reliable in this setting: across more than 500 iterations in 50+ sessions, agents produced fewer than 10 crashed iterations not attributable to external incidents, and these crashes did not derail the corresponding sessions.

Table 2: Generalist agents (**open-prompt**) selecting 10k samples from LLaVA-665K dataset for training LLaVA-1.5-7B models. Scores are averaged over 8 benchmarks; "% of full gain" is the fraction of the base→full-data fine-tuning improvement recovered. Green values are deltas over the best random run. Best outcomes are marked in **bold**. All agents perform better than the best of random while meeting or exceeding evaluated human baselines.

Setting	Avg. score ±std	[min, max]	% of full gain
Base model (no fine-tuning)	28.8	-	0%
Full-data fine-tuning (665k)	37.1	-	100%
Baselines, 10k budget			
Random (baseline, 10 runs)	31.9±0.3	[31.5, **32.5**]	45% (best of 10)
ICONS (Wu et al., 2024)	33.3	-	54%
ARDS (Yang et al., 2025)	33.2	-	52%
Open-prompt Agents, 10k budget, 10 iterations			
Claude Code (Opus 4.7)	**33.7**±0.3 **(+1.2)**	[32.5, **34.0**]	**59%** **(+14%)**
Codex (GPT-5.4)	33.3±0.2 (+0.8)	[33.0, 33.5]	54% (+9%)
Kimi K2.5 (OpenHands)	32.8±0.5 (+0.3)	[32.2, 33.4]	48% (+3%)
Qwen3.5-397B (OpenHands)	33.2±0.2 (+0.7)	[33.0, 33.4]	53% (+8%)

Finding 1: Generalist agents substantially improve training outcomes through data curation, reaching expert-baseline performance in a few iterations. Figure 3 shows that the benefits of open-prompt agentic curation extend beyond the primary LLaVA fine-tuning setting. Across multiple fine-tuning tasks (LLaVA-1.5-7B on LLaVA-665K, SmolVLM-Base on LLaVA-665K, and SmolVLM-Base on Vision-Flan), agents consistently improve over random selection. The same

Table 3: Example open-prompt trajectory on the 10k LLaVA-665K selection task using Claude Code. Scores are averaged over 8 VLM benchmarks after LLaVA-1.5-7B fine-tuning. After an initial subset-balancing policy, most later edits adjust source ratios rather than switch policy families, illustrating the local-exploitation pattern in Finding 2.

#	Score	Status	Policy description	New	Grounded	Effective	*Shallow*
–	28.95	keep	Pre-init baseline, no fine-tuning	–	–	–	–
–	32.38	keep	Random 10k baseline	–	–	–	–
1	33.06	keep	Balanced subset; OCR boost; no text_only	✓	✓	✓	✗
2	32.88	discard	Balanced subsets + longest assistant responses	✓	✗	✗	✓
3	32.06	discard	Balanced subsets + text_only (500)	✗	✗	✗	✓
4	33.23	keep	Equal allocation: 2k per visual subset	✗	✗	✓	✓
5	32.56	discard	Boost VG/GQA; reduce COCO/OCR	✗	✗	✗	✓
6	33.63	keep	Near-equal mix: COCO=2500, others=1875	✗	✓	✓	✗
7	32.95	discard	COCO=3000, others=1750	✗	✓	✗	✗
8	33.70	keep	OCR-boosted mix: COCO=2000, OCR/-Text=2250	✗	✓	✓	✗
9	33.02	discard	Heavy OCR+VG: COCO=1500	✗	✓	✗	✓
10	**33.74**	keep	GQA+TextVQA boost: GQA=2250, TextVQA=2250	✗	✓	✓	✗
Execution success: 10/10			**Trajectory totals**	2/10	6/10	5/10	5/10

pattern also appears in pretraining: on the DataComp Small CLIP-pretraining task, the agent achieves a clear gain over the strongest filtering baseline, top-30% CLIP L/14-score filtering (Brunner, 2023), showing that the agentic curation loop is not limited to instruction-tuning data. Table 2 compares generalist agents with random selection and published data-selection baselines on the primary LLaVA-665K/LLaVA-1.5-7B task. Here, all open-prompt agents improve over the best random run on average. Claude Code performs best, reaching 33.7 versus 32.5 for the best random run and 33.3/33.2 for ICONS (Wu et al., 2024)/ARDS (Yang et al., 2025); this recovers 59% of the full-data gain using only $\approx$ 1.5% of the 665k pool. Codex and Qwen are close to the 10k human-designed baselines, while Kimi improves over random but remains below them.

3.3 Trajectory: Success Comes from Local Heuristics

Finding 2: Open-prompt agents make weakly grounded choices and over-exploit local policy families. Despite favorable final scores, open-prompt agents show two recurring failure modes. First, their stated rationales are often weakly grounded. Run notes typically invoke plausible but generic goals—"filter low-quality examples," "increase diversity," "try different random seeds," or "balance domains"—without tying the next policy to a concrete evidence source, hypothesis, or ablation (Tables 3,25). This produces a form of *vibe optimization*: the plan sounds like data research, but the operational criterion is underspecified.

Second, when the agent commits to a manifest, it usually instantiates the lowest-cost version of the idea: a source-ratio change, response-length threshold, metadata filter, or seed change. These local moves might still improve the score, but they keep the search within the same policy family. Once the agent finds a workable family, it rarely pivots to another, and the final outcome is bounded by that family (Tables 3,5).

4 Scaffolds Convert Execution into Exploration

The previous section shows that open-prompt agents are useful executors but narrow researchers: they can run the loop and improve scores, yet they tend to settle into local data-policy edits. We now introduce a scaffold ladder with different levels of guidance to change this behavior. The key question is whether guidance merely changes what agents write in their plans, or whether it changes the policies they actually execute.

4.1 Scaffold Design

We study two light scaffolds and two heavy scaffolds. The light scaffolds are designed as awareness-level interventions. *Data strategies* gives the agent a short list of possible policy families, including source balancing, quality filtering, diversity sampling, task targeting, deduplication, difficulty scoring, conversation-structure filtering, and image-metadata filtering. *Research papers* gives the agent access to paper-derived skill cards and asks it to consult them when useful. These scaffolds expand the agent's stated option set, but do not require a specific decision protocol.

The heavy scaffolds instead impose a binding re-

Table 4: Scaffold conditions. Light scaffolds provide optional guidance; heavy scaffolds make research discipline part of the task contract.

Condition	Constraint	What the agent must do	Operative prompt language
Open-prompt	None	Run the standard curation loop	Select at most B examples; inspect data, write scripts, submit a manifest, read logs, and iterate.
Light I Data Strategies	Optional	Consider broader policy families	Curation strategies include quality filtering, source balancing, semantic diversity, task balancing, deduplication, and length/format filters.
Light II Research Papers	Optional	Consult prior-method skill cards	List available paper-derived skill cards and **consider using** them when helpful.
Heavy I Self-Research	Binding	Justify each run with evidence	Before editing the policy, write an **observation**, **hypothesis**, expected effect, and minimal change. The observation **must cite concrete evidence**.
Heavy II Adapt Papers	Binding	Adapt a prior method into code	For every non-baseline iteration, **choose a prior method**, explain why it applies, **adapt** it to the available fields and budget, implement the policy, and **validate** the manifest.

search protocol. In *self-research*, each non-baseline iteration must begin with a written observation, hypothesis, expected metric effect, and minimal policy change, grounded in prior runs or evaluation logs. In *adapt papers*, each non-baseline iteration must cite a specific prior method or skill card, explain why it applies, adapt it to the available fields and budget, and validate the resulting manifest before training. The scaffold constrains the structure of the research loop, but not the policy content: the agent still chooses which failure mode to address, which method to adapt, and how to implement the adaptation.

Essentially, light scaffolds test whether making the solution space visible is enough; heavy scaffolds test whether the agent needs a procedural constraint that forces evidence-grounded or method-grounded execution.

4.2 Intervention Outcome

Figure 4 and Table 5 summarize the scaffold results. The main pattern is that awareness-level scaffolds broaden what the agent considers, but protocol-level scaffolds change what the agent executes. Light scaffolds reduce variance and increase some trajectory quality metrics, but they do not materially improve the best outcome over the open-prompt condition. The strongest result comes from the paper-adaptation scaffold: its best 10k policy reaches 34.9 average score, exceeding the open-prompt agent and the evaluated 100k ARDS baseline while using one-tenth as many examples.

Finding 3: Light scaffolds broaden consideration, but do not reliably improve executed policy quality. The data-strategies scaffold increases the rate of new-policy-family moves from 27% to 43%, suggesting that the agent does consider a broader space when policy families are surfaced explicitly. The research-paper scaffold increases grounding from 57% to 70% and reduces shallow moves from 47% to 37%, indicating non-trivial trajectory changes. However, neither light scaffold improves the best outcome beyond the open-prompt maximum of 34.0. The data-strategies condition reaches the same maximum, and the research-paper condition remains below it.

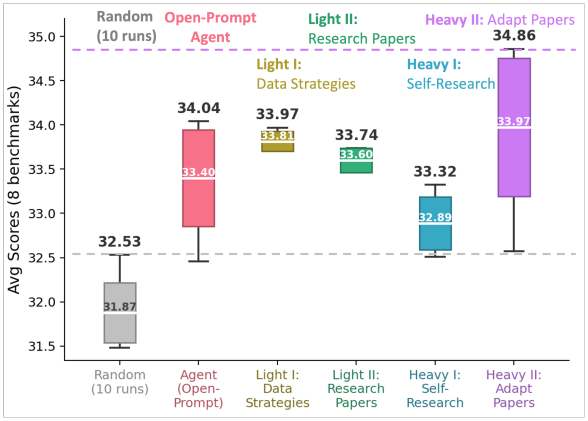


Figure 4: Agentic data-curation results with different scaffolds (Claude Code, 10-iteration). Fine-tuning pre-trained LLaVA-1.5-7B on curated subsets of 10k samples from the LLaVA-665K instruction-tuning pool and evaluating on 8 benchmarks. Scaffolding significantly alters agents' behavior and outcomes. Compared to **open-prompt**, **light** scaffolds reduce outcome variances but do not improve the best outcome. **Heavy** scaffolds change the outcome substantially, but the effects may be for the better or for the worse depending on the specific scaffold design.

The reason is that light scaffolds often change the agent's vocabulary more than its implementa-

Table 5: Effect of scaffold design on Claude Code for 10k-example selection from LLaVA-665K to fine-tune LLaVA-1.5-7B. Outcome scores are averaged over eight benchmarks; trajectory diagnostics report the fraction of iterations with each annotation. Green deltas are relative to open-prompt.

Condition	Outcome			Trajectory diagnostics			
	Mean	Range	Best executed policy	New ↑	Grounded ↑	Effective ↑	Shallow ↓
Open-prompt baseline	*33.7 59% gain*	*[32.5, 34.0]*	Source-ratio tuning: boost GQA and TextVQA	*27%* ±12	*57%* ±6	***30%*** ±20	*47%* ±6
Light scaffolds: optional guidance							
Data Strategies Light I	**33.8 60% gain**	[**33.7**, 34.0]	8k COCO-heavy source mixture	**43%** ±12% (+16%)	51% ±10%	23% ±12%	46% ±5%
Research Papers Light II	33.6	[33.4, 33.7]	Scaled-up source proportions from the best prior run	38% ±6%	**70%** ±11% (+13%)	27% ±9%	**37%** ±19% (-10%)
Heavy scaffolds: binding research protocol							
Self-Research Heavy I	32.9	[32.5, 33.3]	Remove text_only/VG; shift small OCR_VQA quota to COCO	34% ±5%	91% ±10%	23% ±8%	9% ±10%
Adapt Papers Heavy II	**34.0 63% gain**	[32.6, **34.9**] best:+0.9/+11%	**EL2N top-loss selection + assistant-loss noise filtering**	**67%** ±6% (+40%)	**100%** ±0% (+43%)	20% ±0%	**0%** ±0% (-47%)

Non-agent reference points. Random 100k: 33.7 / 59% gain; Random 200k: 34.0 / 63% gain; ICONS 100k: 34.5 / 69% gain (Wu et al., 2024); ARDS 100k: 34.1 / 64% gain (Yang et al., 2025).

tion. The agent names richer policy families, but still tends to instantiate low-cost variants: quota changes, length filters, simple quality proxies, or scaled versions of previously successful mixtures (Table 27,28). This is a rational short-horizon behavior. Once the agent finds a workable family from prior runs, local edits preserve information from prior runs and have a reasonable chance of producing small gains. Light scaffolds make different policy options visible, but do not force the agent to pay the implementation cost.

Finding 4: heavy scaffolds enforce grounding and enable larger breakthroughs. The heavy scaffolds change the executed trajectory more substantially. Self-research increases grounded iterations from 57% to 91% and reduces shallow operations from 47% to 9%. Adapt-papers is stronger: all iterations are grounded, shallow operations are eliminated, and the rate of new-policy-family moves rises to 67%. This indicates that the scaffold is not merely changing the wording of the agent's notes; it changes what the agent actually tries.

The effect is clearest in the best adapt-papers trajectory. After several failed method adaptations, the agent moves into a training-dynamics family: it uses an EL2N-style loss proxy to select high-loss examples, then refines that policy with an assistant-loss noise filter to remove the highest-loss samples likely to reflect incoherent or noisy responses. This produces the best 10k result, 34.9, above the open-prompt agent and the evaluated 100k baselines.

This result also explains why trajectory diagnostics are needed. Heavy scaffolds do not necessarily maximize the number of locally effective iterations. The adapt-papers scaffold has only 20% effective iterations by the immediate-improvement criterion, lower than the open-prompt condition. But the trajectory reaches a policy family that open-prompt search did not reach. For autonomous data research, the important quantity is therefore not only the rate of small local gains, but whether the search reaches higher-upside regions of the policy space.

The scaffold comparison should not be read as showing that heavier is always better. Self-research improves grounding but lowers the final score vs. open-prompt. Heavy scaffolds combine several ingredients—method grounding, structured logging, category rotation, and procedural constraints—so the experiment does not isolate a single causal component. The more conservative conclusion is that awareness alone is insufficient: to convert agent execution into policy exploration, the scaffold must make evidence-grounded or method-grounded action part of the task contract.

5 Broader Investigations

5.1 Scaling the Curation Compute Budget

Finding 5: Longer curation search continues to pay off under a fixed data budget. Figure 5 varies the number of agent iterations from 10 to 50 while keeping the candidate pool, data budget, training recipe, and evaluation suite fixed. The

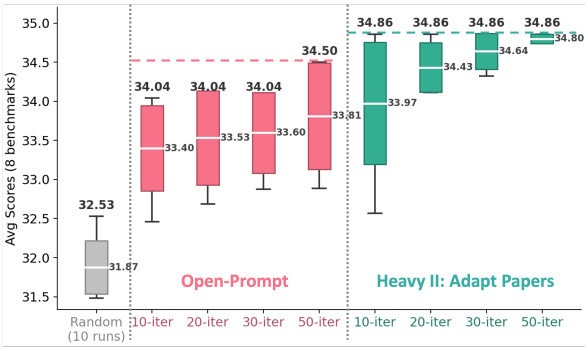


Figure 5: Ablation on the number of iterations (10-50) in each agent session (Claude Code).

result is notable: average performance continues to improve within this range rather than clearly plateauing. Under open-prompting, the gains accumulate gradually, consistent with extended local search over source mixtures and related heuristics. Under the Heavy II scaffold, the best score is already found within the first 10 iterations, but additional iterations reduce variance and move the average upward.

This suggests that the agent iteration budget is itself a meaningful compute axis for data curation. In a finite-data regime, where additional raw data is unavailable, expensive, or lower quality, progress need not come only from training on more examples. Additional compute can instead be spent on searching over data policies: selecting different subsets, testing distinct policy families, adapting prior methods, or rewriting examples. From this perspective, CURATION-BENCH provides a way to study how curation quality scales with search compute under a fixed data pool.

5.2 Rewriting Extension

The selection experiments optimize *which* examples to train on. To test whether CURATION-BENCH also supports richer data actions, we instantiate it as a data-rewriting task: the agent selects examples, rewrites them with an external multimodal large language model (MLLM) tool, verifies the rewritten samples, and submits the transformed dataset through the same validation, contamination, training, and evaluation loop. This setting uses a stricter *self-research*-like scaffold. Every post-baseline iteration must inspect raw evaluation failures, name a failure mode, select source examples tied to that mode, rewrite them with the MLLM, verify the rewritten examples, and pass contamination checks before training. Full rewrite scaffold is provided in code repo. Results (Table 6) show that CURATION-BENCH is not limited to subset selection: the same framework can evaluate richer data actions, such as targeted rewriting, when equipped with appropriate tools.

Table 6: Rewriting results on LLaVA-665K/LLaVA-1.5-7B with self-research-like rewriting scaffolding (Claude Code agent). The agent selects and rewrites 10k examples using an external MLLM tool. Green deltas show improvement over the *Template Rewriting* baseline. Bottom rows provide data-efficiency reference points. The choice of rewriting model substantially affects outcomes; Qwen3.5-9B consistently outperforms smaller alternatives.

Setting	Data size	Avg. score	% of full gain
Base model	0	28.8	0%
Full-data fine-tuning	665k	37.1	100%
Random (10-run max)	10k	32.5	45%
Template Rewriting (baseline) - Wang et al. (2024b)	*10k*	*32.6*	*46%*
Agent (tool: Qwen3.5-2B): 10-iter	10k	33.4 (+0.8)	55% (+9%)
Agent (tool: Qwen3.5-9B): 10-iter	10k	**34.1 (+1.5)**	**64% (+18%)**
Agent (tool: Gemma3-4B): 10-iter	10k	32.8 (+0.2)	48% (+2%)
Agent (tool: Qwen3.5-2B): 15-iter	10k	34.0 (+1.4)	63% (+17%)
Agent (tool: Qwen3.5-9B): 20-iter	10k	**34.7 (+2.1)**	**71% (+25%)**
Agent (tool: Gemma3-4B): 15-iter	10k	33.1 (+0.5)	52% (+6%)
Agent (tool: Gemma3-4B): 25-iter	10k	33.6 (+1.0)	58% (+12%)
Random 50k	50k	33.4±0.2	55%±3%
Random 100k	100k	33.7±0.2	59%±3%
Template Rewriting 50k	50k	33.5	57%
Template Rewriting 100k	100k	33.8	60%

5.3 Comparisons with Broader Data Research

In addition, we compare prominent data selection from existing research for a broader comparison with the agents. Different from experiments in Section 3, which compares with human-designed methods originally proposed for these tasks, we now expand the comparision and include broader methods that were originally proposed for generic use cases or different applications. All these methods are implemented from scratch, adapted for new tasks and targeting the same validation goals as the data agents.

We compare with representative approaches, LESS (Xia et al., 2024), DSIR (Xie et al., 2023b), and D4 (Tirumala et al., 2023), to achieve a good coverage of distinct mythologies. Specifically, LESS (Xia et al., 2024) selects instruction tuning data via estimated influence on the validation tasks. It requires first running an expensive training session to evaluate the influence of every candidate example on the target tasks, which scales with the size of the total candidate data. DSIR (Xie et al., 2023b) selects data for pre-training/continued pre-training by matching the distribution to the target tasks. It operates on the n-gram distribution and selects data directly targeting validation tasks. D4 (Tirumala et al., 2023) conducts density-based deduplication for pre-training-scale data filtering/selection. This method works independently of the validation tasks. We conduct 3 independent runs for each of these

methods (selection, training, evaluation) and report their results on selecting from LLaVA-665K to fine-tune LLaVA-1.5-7B and DataComp Small benchmark (Filtering track) in Table 7 and 8, respectively.

Table 7: Comparisons with broader data selection methods (LESS, DSIR, D4) on selecting 10k samples from LLaVA-665K to fine-tune LLaVA-1.5-7B, evaluated on 8 benchmarks. "% of full gain" is the fraction of the base→full-data fine-tuning improvement recovered. Green values denote deltas over the random baseline (37%) by more than 10% in "% of full gain". Best outcomes in each catgory are marked in **bold**.

Setting	Avg. score ±std	% of full gain
Base model (no fine-tuning)	28.8	0%
Full-data fine-tuning (665k)	37.1	100%
Baselines, 10k budget		
Random (baseline, 10 runs)	31.9±0.3	37%
ICONS	33.3	54% (+17%)
ARDS	33.2	53% (+16%)
LESS	**33.6**±0.3 **(+1.7)**	**58% (+20%)**
DSIR	32.5±0.2 (+0.6)	45% (+8%)
D4	32.0±0.3 (+0.1)	39% (+2%)
Open-prompt Agents, 10k budget, 10 iterations		
Claude Code (Opus 4.7)	**33.7**±0.3 **(+1.8)**	**59% (+22%)**
Codex (GPT-5.4)	33.3±0.2 (+1.4)	54% (+17%)
Kimi K2.5 (OpenHands)	32.8±0.5 (+0.9)	48% (+11%)
Qwen3.5-397B (OpenHands)	33.2±0.2 (+1.3)	53% (+16%)
Claude+Scaffolds, 10k budget, 10 iterations		
Light I, Data Strategies	33.8±0.1 (+1.9)	60% (+23%)
Light II, Research Papers	33.6±0.1 (+1.7)	58% (+20%)
Heavy I, Self-Research	32.9±0.3 (+1.0)	49% (+12%)
Heavy II, Adapt Papers	**34.0**±0.8 **(+2.1)**	**63% (+26%)**

Table 8: Comparisons with broader data selection methods (DSIR, D4) on the DataComp Small benchmark (Filtering track). We pre-train ViT-B/32 CLIP models and report average scores across 38 tasks. We ran DSIR and D4 each for 3 different filtering budgets (retaining 70%/50%/30% data). % of "Full data" is the fraction of the achieved performance relative to the score from training on full data. Green values denote improvements over the Random 50% baseline. Best outcomes in each category are marked in **bold**.

Setting	Avg. score ±std	% of "Full data"
Full data	0.134±0.001	100%
Baselines, 10 iterations		
Random 50% data	0.125±0.001	93%
CLIP score filtering (L/14-top 30% data, best DataComp baseline)	**0.147**±0.003 **(+0.022)**	**110% (+17%)**
DSIR (retaining 70% data)	0.125±0.002	93%
DSIR (retaining 50% data)	0.123±0.003	92%
DSIR (retaining 30% data)	0.108±0.005	81%
D4 (retaining 70% data)	**0.131**±0.002 **(+0.006)**	**98% (+5%)**
D4 (retaining 50% data)	0.125±0.001	93%
D4 (retaining 30% data)	0.109±0.002	81%
Open-prompt Agents, 10k budget, 10 iterations		
Claude Code (Opus 4.7)	**0.166**±0.004 **(+0.041)**	**124% (+31%)**

These methods are implemented directly in the same scenario and targeting the same validation tasks. LESS consumes 10x GPU hours for selection compared to the final training, spending the same amount of compute as 10 agent iterations. For the DataComp/CLIP experiment, DSIR and D4 each select for 3 different budgets to search for their best threshold. With all the confounding factors excluded, the open-prompt agent still reaches a performance comparable to the best human baselines, while the scaffold Heavy II, Adapt Papers could further surpass it.

Besides, the implementation of such human-designed methods is often highly specific — adapting them to a different task is nontrivial, and their effectiveness in extended scenarios is unknown a prior. This highlights a unique benefit and a major advantage of the data agents–these generalist agents provide an automated pipeline rather than a fixed solution. The same agent can be plugged into entirely different scenarios with minimal modifications. Beyond the data selection task for multimodal instruction tuning, the same agents work comparably well in distinct tasks, such as rewriting or selection for DataComp/CLIP training.

6 Conclusion

This paper introduced CURATION-BENCH, a benchmark for evaluating whether generalist agents can conduct training-data curation as an iterative policy-search process. By fixing the model, training recipe, and evaluation suite, CURATION-BENCH isolates the curated data policy as the object of optimization and records the full trajectory of agent decisions.

Our results show both promise and a limitation. Open-prompt agents are already useful data-curation executors: they produce valid datasets, run the training/evaluation loop, and match strong human-designed selection baselines at small data budgets. However, trajectory diagnostics reveal an execution–research gap: agents often improve scores through local heuristic edits rather than by exploring distinct policy families. Scaffold design is the central lever we identify for narrowing this gap. Awareness-level scaffolds broaden plans but do not reliably change executed policies; method-adaptation scaffolds force agents to operationalize prior work and can open higher-upside regions of the data-policy space.

These findings suggest that future evaluations of autonomous data research should report not only final scores, but also scaffold strength and trajectory diagnostics: open-ended instructions and paper-grounded protocols answer different scientific questions.

Limitations

Our experiments instantiate CURATION-BENCH primarily in vision-language instruction tuning, with one smaller CLIP-style pretraining instantiation and a rewriting extension. The conclusions may differ for pretraining mixtures, code, math, domain-specific instruction tuning, or settings where evaluation is less reliable. Our scaffold comparison is also not a full factorial study: heavy scaffolds combine method grounding, structured logging, and stronger procedural constraints, so we identify a useful design pattern rather than isolating a single causal ingredient. Finally, trajectory labels require annotation; although they make agent behavior more legible, they introduce their own judgment calls and should be reported with rubrics and examples.

Broader Impacts

Agentic data curation could reduce the cost of fine-tuning by identifying smaller, higher-value subsets, but it also increases the need for oversight. Agent-curated datasets may amplify biases, remove underrepresented examples, or optimize toward narrow evaluation targets if deployed without review. Because paper-grounded scaffolds lower the engineering barrier for data optimization, they could also accelerate harmful applications. We therefore recommend treating agent-curated datasets as candidate artifacts subject to human or programmatic review, reporting scaffold strength and trajectory diagnostics alongside outcome scores, and retaining contamination checks and held-out evaluations.

Acknowledgment

Ruoxi Jia and the ReDS lab acknowledge support from the National Science Foundation through grants IIS-2312794, IIS-2313130, OAC-2239622, CNS-2424127, OAC-2613761, and the Amazon-Virginia Tech Initiative for Efficient and Robust Machine Learning. We also acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper.

References

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*.

Anthropic. 2025. Claude code [computer software]. https://docs.anthropic.com/en/docs/claude-code.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Eshta Bhardwaj, Harshit Gujral, Siyi Wu, Ciara Zogheib, Tegan Maharaj, and Christoph Becker. 2024. Machine learning data practices through a data curation lens: An evaluation framework. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 1055–1067.

Dustin Brunner. 2023. Datacomp challenge.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, and 1 others. 2024. Mle-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and 1 others. 2024. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087.

Mayee F Chen, Tyler Murray, David Heineman, Matt Jordan, Hannaneh Hajishirzi, Christopher Ré, Luca Soldaini, and Kyle Lo. 2026. Olmix: A framework for data mixing throughout lm development. *arXiv preprint arXiv:2602.12237*.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, and 1 others. 2024. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 11198–11201.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. 2024. Doge: domain reweighting with generalization estimation. In *Proceedings of the 41st International Conference on Machine Learning*, pages 12895–12915.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah Pratt, Vivek Ramanujan, Yonatan

Bitton, Kalyani S. Marathe, Stephen O. Mussmann, Richard Vencu, and 15 others. 2023. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 36:27092–27112.

Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR.

Sachin Goyal, Pratyush Maini, Zachary Chase Lipton, Aditi Raghunathan, and J Zico Kolter. 2024. The science of data filtering: Data curation cannot be compute agnostic. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, and 1 others. 2024. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14375–14385.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. *Computer software]. Hugging Face. https://github. com/huggingface/accelerate*.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2023. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. 2024. Dsbench: How far are data science agents from becoming data science experts? *arXiv preprint arXiv:2409.07703*.

Feiyang Kang, Newsha Ardalani, Michael Kuchnik, Youssef Emad, Mostafa Elhoushi, Shubhabrata Sengupta, Shang-Wen Li, Ramya Raghavendra, Ruoxi Jia, and Carole-Jean Wu. 2025a. Demystifying synthetic data in llm pre-training: A systematic study of scaling laws, benefits, and pitfalls. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10750–10769.

Feiyang Kang, Nadine Chang, Maying Shen, Marc T Law, Rafid Mahmood, Ruoxi Jia, and Jose M Alvarez. 2025b. Adadedup: Adaptive hybrid data pruning for efficient large-scale object detection training. *arXiv preprint arXiv:2507.00049*.

Feiyang Kang, Hoang Anh Just, Anit Kumar Sahu, and Ruoxi Jia. 2023. Performance scaling via optimal transport: Enabling data selection from partially revealed sources. *Advances in Neural Information Processing Systems*, 36:61341–61363.

Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia. 2025c. Autoscale: Scale-aware data mixing for pre-training LLMs. In *Second Conference on Language Modeling (COLM)*.

Andrej Karpathy. 2026. autoresearch: Ai agents running research on single-gpu nanochat training automatically. https://github.com/karpathy/autoresearch. Accessed: 2026-04-29.

Konwoo Kim, Suhas Kotha, Yejin Choi, Tatsunori Hashimoto, Nick Haber, and Percy Liang. 2026. Data-efficient pre-training by scaling synthetic megadocs. *arXiv preprint arXiv:2603.18534*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Patrick Tser Jern Kon, Qiuyi Ding, Jiachen Liu, Xinyi Zhu, Jingjia Peng, Jiarong Xing, Yibo Huang, Yiming Qiu, Jayanth Srinivasa, Myungjin Lee, Mosharaf Chowdhury, Matei Zaharia, and Ang Chen. 2026. Exp-bench: Can ai conduct ai research experiments? In *The Fourteenth International Conference on Learning Representations (ICLR)*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, and 1 others. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen

Men, Kejuan Yang, and 1 others. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, and 1 others. 2024b. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024c. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

Mehran Maghoumi and Summer Liu. 2024. Scale and curate high-quality datasets for LLM training with NVIDIA NeMo curator. NVIDIA Technical Blog. Accessed: 2026-05-07.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14044–14072.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, and 1 others. 2025. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, and 1 others. 2026. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. 2023. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376.

Andrew Ng. 2021. A chat with andrew on mlops: From model-centric to data-centric ai. YouTube video, DeepLearning.AI. Accessed: 2026-05-05.

Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411.

NVIDIA. 2024. NeMo microservices. NVIDIA NGC Catalog. A comprehensive suite of microservices for fine-tuning, evaluating, and serving large language models on Kubernetes. Accessed: 2026-05-07.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607.

Mykola Pinchuk. 2026. Tml-bench: Benchmark for data science agents on tabular ml tasks. *arXiv preprint arXiv:2603.05764*.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. 2026. Posttrainbench: Can llm agents automate llm post-training? *arXiv preprint arXiv:2603.08640*.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents. *URL: https://qwen.ai/blog?id=qwen3.5*.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. 2023. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36:53983–53995.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024a. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Shijian Wang, Linxin Song, Jieyu Zhang, Ryotaro Shimizu, Ao Luo, Li Yao, Cunjian Chen, Julian

McAuley, and Hanqian Wu. 2024b. Template matters: Understanding the role of instruction templates in multimodal language model evaluation and training. *arXiv preprint arXiv:2412.08307*.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2025. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pages 65882–65919.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Xindi Wu, Mengzhou Xia, Rulin Shao, Zhiwei Deng, Pang Wei Koh, and Olga Russakovsky. 2024. Icons: Influence consensus for vision-language data selection. *arXiv preprint arXiv:2501.00654*.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023a. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. 2023b. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36:34201–34227.

Zhiyang Xu, Chao Feng, Rulin Shao, Trevor Ashby, Ying Shen, Di Jin, Yu Cheng, Qifan Wang, and Lifu Huang. 2024. Vision-flan: Scaling human-labeled tasks in visual instruction tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15271–15342.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Xu Yang, Chen Liu, and Ying Wei. 2025. Data selection matters: Towards robust instruction tuning of large multimodal models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. 2025. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *International Conference on Learning Representations*, volume 2025, pages 82263–82287.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, and 1 others. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9556–9567.

Dylan Zhang, Qirun Dai, and Hao Peng. 2025. The best instruction-tuning data are those that fit. *arXiv preprint arXiv:2502.04194*.

Yuge Zhang, Qiyang Jiang, XingyuHan XingyuHan, Nan Chen, Yuqing Yang, and Kan Ren. 2024. Benchmarking data science agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5677–5700.

Appendices

A Extended Related Work **15**

B Details for CURATION-BENCH **16**
B.1 Source data and models. 16
B.2 Contamination audit 17
B.3 Submission validation 17
B.4 Training 17
B.5 Evaluation 19

C Scaffolds (open-prompt, light, heavy) **19**
C.1 Common Failures for Open-Prompt Agents 19
C.2 Light Scaffolds 19
C.3 Heavy Scaffolds 21
C.4 Rewriting Scaffolds 22

D Experimental Details and Additional Results **23**
D.1 Terminal Agents 23
D.2 DataComp Filtering Setup 23
D.3 Shared Training and Evaluation Protocol 23
D.4 Non-Agent Baseline Construction 24
D.5 Additional Results and Ablation Studies 25
D.6 Example Agent Traces and Trajectory Annotations 26

E Extended Discussion **32**
E.1 From autonomous execution to agent-assisted research 32
E.2 Scaffolds as research artifacts 32
E.3 The autonomy–performance trade-off 32

F Broader Impacts and Considerations **32**
F.1 Broader Societal Impacts 32
F.2 Considerations and Safeguards 33
F.3 Released Artifacts and Residual Risk 33
F.4 Declaration of LLM Use 33

G Prompts and Skill-Card Templates **34**

A Extended Related Work

Autonomous research and machine-learning agents. A growing line of work evaluates language-model agents in interactive environments where they read files, edit code, run commands, and observe feedback. AgentBench introduced a broad multi-environment benchmark for LLM agents (Liu et al., 2023b); MLAgentBench and MLE-bench evaluate machine-learning experimentation and Kaggle-style engineering (Huang et al., 2023; Chan et al., 2024); PostTrainBench evaluates whether CLI agents can post-train base LLMs under bounded compute (Rank et al., 2026); and Terminal-Bench broadens the setting to general command-line tasks (Merrill et al., 2026). AutoResearch and AI-Scientist-style systems push further, placing agents in iterative loops of idea generation, code modification, experiment execution, and reporting (Karpathy, 2026; Lu et al., 2024; Yamada et al., 2025). Conceptually, the AutoResearch framework extends to optimizing training data in the same way as for model and training configurations; preliminary discussion appears in Karpathy (2026), but a systematic study on the data front is absent. CURATION-BENCH fills that gap by isolating training-data curation as the central object of study.

Data-centric AI and data selection. Data-centric AI emphasizes dataset quality, coverage, and labeling as first-class levers for model improvement (Ng, 2021). The literature spans data pruning, coreset selection, influence functions, data valuation, quality filtering, and training-dynamics metrics, etc. (Koh and Liang, 2017; Ghorbani and Zou, 2019; Paul et al., 2021; Sorscher et al., 2022; Northcutt et al., 2021). EL2N and GraNd identify important examples early in training (Paul et al., 2021); instruction-tuning data selection has been studied via quality scoring, diversity, influence approximation, and targeted selection (Xia et al., 2024); and Self-Instruct-style pipelines show that LLMs can also generate instruction data that requires its own filtering and auditing (Wang et al., 2023). Recent works also include scaling laws components that allows efficiently optimizing large-scale training data (e.g., LLM pre-training) with experiments at much smaller scales (Kang et al., 2023, 2025c; Ye et al., 2025). Our work is motivated by this literature but targets a different question: a useful *data agent* must not just execute a known method–it must decide which method family is appropriate, implement it under budget, inspect evidence, and revise the policy.

Data-science agents. Data-science benchmarks evaluate agents that analyze datasets, write code, produce notebooks, and build tabular models. DSEval covers the data-science lifecycle (Zhang et al., 2024); DSBench draws realistic analysis and modeling tasks from sources such as Kaggle and ModelOff (Jing et al., 2024); and TML-Bench evaluates autonomous agents on tabular ML tasks under time limits (Pinchuk, 2026). These targets are valuable but are usually *analysis* or *predictive modeling*; the artifact being optimized is a model or report, not a training corpus. CURATION-BENCH instead measures whether agents can produce a *data policy* whose value is measured by downstream model behavior on generative-model training.

Agent benchmarks. Agent benchmarks such as AgentBench, MLAgentBench, MLE-bench, Post-TrainBench, Terminal-Bench, AutoResearch, and AI-Scientist-style systems evaluate whether agents can operate in interactive environments, modify code, run experiments, and report results (Liu et al., 2023b; Huang et al., 2023; Chan et al., 2024; Rank et al., 2026; Merrill et al., 2026; Karpathy, 2026; Lu et al., 2024; Yamada et al., 2025). Data-centric AI and data-selection research provide many non-agentic methods for pruning, scoring, filtering, and valuing examples (Ng, 2021; Koh and Liang, 2017; Ghorbani and Zou, 2019; Paul et al., 2021; Sorscher et al., 2022; Northcutt et al., 2021; Xia et al., 2024; Wang et al., 2023). Data-science-agent benchmarks evaluate analysis, notebook construction, and tabular modeling rather than the downstream value of a curated training corpus (Zhang et al., 2024; Jing et al., 2024; Pinchuk, 2026). CURATION-BENCH fills the gap summarized in Table 9: it evaluates whether generalist agents can discover, implement, and revise data policies whose value is measured by downstream model behavior.

Table 9: Positioning relative to agent and data-centric benchmarks. The table intentionally focuses on whether the benchmark makes training-data curation the central research object.

Work	Primary target	Agent actions	Data tasks?	Gap for autonomous data research
AutoResearch (Karpathy, 2026)	Training-recipe, code, and hyperparameter optimization	Modify code/configs, run experiments, report	No	Optimizes a pipeline under a fixed data distribution; CURATION-BENCH isolates the data distribution itself as the object of optimization.
AI Scientist (Lu et al., 2024; Yamada et al., 2025)	Automated research loops	Generate ideas, modify code, executes experiments, write paper	Partial	Data side is usually one of many levers, not isolated as the benchmark target.
MLAgentBench (Huang et al., 2023)	ML experimentation	Train models, edit code, inspect output	Partial	Evaluates broad ML experimentation rather than data policy discovery.
MLE-bench (Chan et al., 2024)	ML engineering competitions	Kaggle-style engineering	Partial	Data preparation matters, but objective is competition performance, not data curation mechanisms.
PostTrainBench (Rank et al., 2026)	Autonomous LLM post-training	Fine-tune, curate data, evaluate	Partial	Evaluates end-to-end post-training; our focus is trace-level diagnosis and scaffolds for data strategy exploration.
DSEval (Zhang et al., 2024) / DSBench (Jing et al., 2024) / TML-Bench (Pinchuk, 2026)	Data science and tabular ML	Analyze data, produce notebooks, train tabular models	No	Data is an input to analysis/modeling rather than the artifact being optimized.
Data-centric AI / data selection (Ng, 2021; Paul et al., 2021; Sorscher et al., 2022)	Dataset engineering	Usually non-agentic methods	Yes	Provides methods, but not an agent benchmark for autonomous data research.
CURATION-BENCH (*this work*)	Training data curation	Select, mix, filter, rewrite, inspect, train, evaluate	Yes	Measures whether generalist agents can discover and execute data policies.

B Details for CURATION-BENCH

B.1 Source data and models.

The benchmark draws training pools from two public visual-instruction mixtures and fine-tunes one of four base VLMs.

LLaVA-665K. Following Liu et al. (2024a), we use the `llava_v1_5_mix665k.json` mixture, materialized locally as a single Arrow dataset of 665,298 rows with three columns (`images`, `texts`, `source_subset`). Rows are tagged by their originating `source_subset`, partitioning the pool into six groups: `coco` (364,100), `vg` (86,417), `ocr_vqa` (80,000), `gqa` (72,140), `text_only` (40,688, image-free ShareGPT turns), and `textvqa` (21,953).

Vision-Flan. We additionally use the human-annotated Vision-Flan (Xu et al., 2024) mixture, materialized as a single Arrow dataset with three columns (images, texts, source_subset). It has over 191 task types (approximately 1,000 samples per task, following the released `annotation_191-task_1k` layout).

Models. Each task pins a single `model_key` so that fine-tuning always starts from the same pre-curation checkpoint:

- **LLaVA-1.5-7B** (Liu et al., 2024a) – visual instruction-tuning stage; we full-finetune the language model with the vision tower frozen. Checkpoint released with the artifact as `llava-1.5-7b-init`.
- **Qwen2-VL-2B** (base) (Wang et al., 2024a) – visual instruction-tuning stage; we full-finetune the entire model (vision + encoder + projector merger + language model). Checkpoint can be found in `Qwen2-VL-2B`.
- **Qwen2.5-VL-3B-Instruct** (Bai et al., 2025) – post-training stage starting from the released instruction-tuned checkpoint (`Qwen2.5-VL-3B-Instruct`). We full-finetune the entire model (visual encoder + projector merger + language model), with no module frozen.
- **SmolVLM-2.2B-Base** (Marafioti et al., 2025) – visual instruction-tuning stage starting from the pre-instruct base checkpoint; we full-finetune the language model and the projector with the SigLIP vision tower frozen. Checkpoint can be found in `SmolVLM-Base`.

DataComp-Small. For the CLIP pretraining instantiation we use the *Small* configuration of DataComp (Gadre et al., 2023), the 12.8M-pair filtering-track pool. After downloading we recovered 8.77M of the 12.8M pairs (68.5% of the original pool, the remainder lost to link rot) and resized all images to 256 before storage. The filtering-track rules fix the candidate pool, training compute budget, model architecture, and evaluation suite; the agent's only degree of freedom is the subset of pairs retained for training.

B.2 Contamination audit

Before fine-tuning, an agent may run a contamination audit on its candidate submission to check for overlap with the eight target benchmarks of CURATION-BENCH (HallusionBench, LLaVABench, MMBench, MMMU_DEV_VAL, MMStar, MMVet, MathVista_MINI, OCRBench). We build a normalized question–answer string for each conversation turn in the submission and for each evaluation Q/A pair (lowercase, punctuation stripped, whitespace collapsed), and check (i) exact matches via SHA-256 hashing and (ii) near-duplicates via word 8-gram overlap with similarity ≥ 0.8. The audit aggregates per-benchmark exact-match rate r_e and near-duplicate rate r_n and returns HIGH_RISK ($r_e \geq 0.10$ or $r_n \geq 0.20$), WARNING ($r_e \geq 0.05$ or $r_n \geq 0.10$), or CLEAN otherwise. A HIGH_RISK status forces the agent to revise its selection and re-audit before submission can proceed; WARNING does not block submission but reports the overlap to alert the agent; CLEAN proceeds normally.

B.3 Submission validation

Submission is the gate that pins a candidate dataset as the input to the training stage. We enforce two conditions: (i) **schema**: the path is loadable as a HuggingFace dataset with columns `images` and `texts`; (ii) **budget**: the row count equals the task's `target_rows`; A validated submission is archived into the run directory and any prior fine-tuning result for the iteration is invalidated, so submission acts as a commit point: the agent may iterate freely before calling submit, but once accepted the dataset is frozen for the rest of the iteration.

B.4 Training

CURATION-BENCH fixes the entire fine-tuning stage so that the only variable across an agent's iterations is the curated subset. Given a submitted subset, the harness fine-tunes the task's target model on a single NVIDIA A100 GPU for one epoch under a fixed recipe; no training hyperparameter is exposed to the agent. Each target model in the suite has its own recipe. For LLaVA-1.5-7B and SmolVLM-Base, the vision tower is frozen and only the language model and the multi-modal projector are updated; the detailed training configurations can be found in Table 10 (LLaVA-1.5-7B, except for the Training hardware) and Table 13 (SmolVLM-Base). For Qwen2-VL-2B and Qwen2.5-VL-3B-Instruct, the entire model is full-finetuned with FlashAttention-2 (Tables 11 and 12).

Table 10: Fixed fine-tuning recipe used for all agent and baseline submissions.

Component	Value
Initial model	LLaVA-1.5-7B init (`HF checkpoint`)
Training hardware	1 NVIDIA A100 GPU per model
Training epochs	1
Vision tower	Frozen
Per-device batch size	1
Gradient accumulation	16 for one-GPU subset runs
Learning rate	2×10^{-5}
Learning-rate schedule	Cosine
Warmup ratio	0.03
Precision	bfloat16
Optimizer	Fused AdamW
Gradient checkpointing	Enabled
Formatting	LLaVA chat-template normalization
Loss masking	Assistant-token masking

Table 11: Fine-tuning settings for Qwen2-VL-2B in CURATION-BENCH.

Component	Value
Initial model	Qwen2-VL-2B
Training hardware	1 NVIDIA A100 GPU
Training epochs	1
Trainable modules	Entire model
Per-device batch size	1
Gradient accumulation	8
Learning rate	1×10^{-5}
Learning-rate schedule	Cosine
Warmup ratio	0.03
Weight decay	0.1
AdamW β_2	0.95
Precision	bfloat16
Optimizer	adamw_torch_fused
Attention implementation	FlashAttention-2
Gradient checkpointing	Enabled

Table 12: Fine-tuning settings for Qwen2.5-VL-3B-Instruct in CURATION-BENCH.

Component	Value
Initial model	Qwen2.5-VL-3B-Instruct
Training hardware	1 NVIDIA A100 GPU
Training stage	Post-training
Training epochs	1
Trainable modules	Entire model
Per-device batch size	1
Gradient accumulation	8
Learning rate	1×10^{-5}
Learning-rate schedule	Cosine
Warmup ratio	0.03
Weight decay	0.1
AdamW β_2	0.95
Precision	bfloat16
Optimizer	adamw_torch_fused
Attention implementation	FlashAttention-2
Gradient checkpointing	Enabled

Table 13: Fine-tuning settings for SmolVLM-Base (2.2B) in CURATION-BENCH.

Component	Value
Initial model	SmolVLM-2.2B-Base
Training hardware	1 NVIDIA A100 GPU
Training epochs	1
Trainable modules	Language model + multi-modal projector (vision tower frozen)
Per-device batch size	16
Gradient accumulation	1
Learning rate	1×10^{-5}
Learning-rate schedule	Cosine
Warmup ratio	0.03
Weight decay	0.0
Precision	bfloat16
Optimizer	adamw_torch_fused
Attention implementation	SDPA
Gradient checkpointing	Enabled

For the DataComp instantiation, the harness follows the published DataComp-Small filtering-track recipe verbatim (Table 14): a CLIP ViT-B/32 is trained from scratch on the agent's filtered subset using OpenCLIP, with total training compute fixed at 12.8M processed image–text pairs regardless of the size of the filtered subset, so a full-pool run is $\sim$1.5 passes over the 8.77M available pairs (via the framework's with-replacement WebDataset sampling). All hyperparameters are the Small-track defaults reported by Gadre et al. (2023) and instantiated in the `mlfoundations/datacomp` repository; no training hyperparameter is exposed to the agent.

Table 14: CLIP pretraining settings for the DataComp instantiation (Small filtering track). All values are the published Small-track defaults from Gadre et al. (2023) (DataComp `scale_configs.py`) and OpenCLIP defaults inherited through DataComp's `train.py` driver. The only setup choice particular to our hardware is running the global batch on a single H100; the DataComp default global batch size is also 4096, so no accumulation or per-GPU rescaling is required.

Component	**Value**
Initial model	CLIP ViT-B/32 (random init)
Training framework	OpenCLIP (DataComp `train.py` driver)
Training hardware	1 NVIDIA H100 GPU
Image storage resolution	256×256 (pre-downloaded)
Model input resolution	224×224 (OpenCLIP ViT-B/32)
Total samples seen	12,800,000 image–text pairs (Small-track budget)
Data sampling	WebDataset, with replacement (`-dataset-resampled`)
Global batch size	4096
Optimizer	AdamW
Learning rate	5×10^{-4}
LR schedule	Cosine decay
Warmup steps	500
Weight decay	0.2
AdamW (β_1, β_2)	$(0.9,\ 0.98)$
AdamW ϵ	1×10^{-6}
Precision	Automatic mixed precision (fp16)
Gradient checkpointing	Enabled
Loss	Local-loss with `-gather-with-grad`
Wall-clock per run	$\sim$2 hours training + $\sim$2 hours evaluation

B.5 Evaluation

The benchmark fixes the evaluation stage and exposes no evaluation parameter to the agent. The protocol – VLMEvalKit wrapper, the eight target benchmarks, the shared Qwen3.5-27B judge endpoint (Table 18), and the aggregation rule (Table 19) – is exactly the one described in Section D.3.

For the DataComp instantiation, the harness uses the standard DataComp evaluation suite (Gadre et al., 2023): 38 zero-shot tasks spanning ImageNet and its distribution shifts, VTAB classification tasks, retrieval (Flickr30k, MSCOCO), and additional classification benchmarks. We report the unweighted mean accuracy across the 38 tasks, matching the filtering-track leaderboard. Evaluation takes $\sim$2 hours per run on a single H100.

C Scaffolds (open-prompt, light, heavy)

C.1 Common Failures for Open-Prompt Agents

Table 15 summarizes common failure modes of open-prompt generalist agents in data curation tasks. Table 16 provides a non-exhaustive taxonomy of data-curation strategy families. The families are partially disconnected: moving from one to another often requires new tools, proxies, and assumptions.

C.2 Light Scaffolds

The design principle of our light scaffolds is to remain task-agnostic so that the same prompt could plausibly ship as a default for any data-curation task. The goal of this paper is not to add another data-curation strategy but to make ready-to-use generalist agents practical for frontline practitioners. We explore two strategies that best embody that ideology: (I) providing a list of curation strategies organized

Table 15: Common failure modes of open-prompt generalist agents on data curation tasks.

Failure mode	Description	Potential scaffold
Only doing heuristics	Agent repeatedly applies length, metadata, or score filters with weak justification.	Require policy-family novelty or evidence-backed ablations.
Vibe optimization	Agent states that data is "higher quality" without measurement.	Data audit reports, quality classifiers, and hypothesis templates.
Optimizing random seeds	Agent attributes noisy gains to data policy or spends budget on seeds.	Fixed seed protocols and paired comparisons.
Not using provided skills	Agent receives a skill list but never calls or implements the skills.	Skill cards with executable entry points and mandatory decision checkpoints.
No original research	Agent repeats known defaults and does not propose new data mechanisms.	Force comparison across policy families before local search.
Shallow eval-log reading	Agent reads aggregate score but not raw failures.	Structured error extraction, clustering, and traceable rewrite plans.
Overfitting visible evals	Agent creates examples too close to evaluation outputs.	Contamination checks, hidden evals, and red-team audits.
Heavy scaffold contradiction	Detailed pipelines improve scores but reduce autonomy.	Report scaffold strength as an experimental variable.
(rewriting) Selection without rewriting	Agent is told to rewrite but only selects existing examples.	Separate tools and budgets for rewrite generation and validation.

Table 16: A non-exhaustive taxonomy of data-curation strategy families. The families are partially disconnected: moving from one to another often requires new tools, proxies, and assumptions (Bhardwaj et al., 2024; Maghoumi and Liu, 2024; NVIDIA, 2024).

Strategy family	Example interventions	Required machinery
Quality and format	Remove malformed, low-quality, or unsafe examples	Validators, quality models, metadata
Distribution/domain	Tune source and domain mixture ratios	Domain labels, mixture search
Semantic coverage	Cluster examples, maximize diversity, cover skills	Embeddings, clustering, facility-location objectives
Task/instruction level	Balance task types, tools, reasoning modes	Task classifiers, taxonomies
Training dynamics	Select by loss, EL2N, gradient norm, forgetting	Warm-up training, per-example statistics
Dedup/contamination	Remove near-duplicates and benchmark leakage	MinHash, embedding similarity, benchmark filters
Style and response form (*rewriting*)	Rewrite for concise, tool-using, or stepwise style	Rewriting model, validators
Generation feedback (*data synthesis*)	Add data from model failures or self-play	Eval logs, error classifiers, synthetic generation

by policy family and instructing the agent to explore them, and (II) compiling research papers on data tasks into skill cards for the agent to reference.

I. Concise instructions toward desired trajectories. To counter the agent's tendency to fixate on a single strategy family, we surface a list of diverse operations upfront and prompt the agent to explore beyond them. The instruction is deliberately framed at the level of policy families rather than task-specific recipes: this expands the agent's awareness of the solution space without prescribing implementations and preserves autonomy over *how* each idea is instantiated.

II. Research papers as skill cards. The value of human data experts lies in knowing which methods are viable for a given task and when each is most likely to help. Rather than hand-injecting that expertise per task, we compile relevant research papers into reusable skill cards that the agent can consult on its own. Concretely, we prompt Claude Code with an automated pipeline (Appendix G) to compile a library of skill cards from VLM data-curation papers, each formatted as a decision-first cookbook entry rather than a paper summary. The compilation is offline–skill cards are prepared before any agent run–and the agent prompt is augmented with two short notes: list all available skill cards at the start of each session, and *use the reference skills when helpful*.

Skill cards compiled from research papers on *data curation vision-language models (VLMs)*

Characteristics of the skills

– Four top-level buckets, each a stage of the VLM data pipeline: data-acquisition, data-curation, data-selection, data-synthesis
– Each skill is one folder containing exactly one file: ‘SKILL.md‘ with a title that combines the paper’s short title and source.
– 1 paper -> 1 skill. No multi-paper aggregations; no extra assets (no code, no figures, no PDFs).

Per-file structure (rigidly templated)

Tone / philosophy
– Decision-first ("Use this skill when. . . " / "Avoid it when. . . ") rather than paper summary.
– Procedures are skeletons — the *shape* of the method, not full reproduction. Each step ends with "See paper for details."
– Knowledge-cards optimized for a downstream agent to retrieve and act on, not for human reading. They look like cookbook recipe cards keyed by paper.

Common Failures under Light Scaffolding. Table 17 summarizes the failures: light interventions improve the written plan but rarely the executed trajectory. The agent acknowledges the broader options and then chooses the easiest implementable heuristic. The bottleneck is not knowledge–the agent often *names* training dynamics, semantic clustering, or rewriting–but the conversion of a high-level idea into a valid, budget-aware implementation with the right diagnostics. This is the recurrent pattern: current generalist agents are strong executors but weak autonomous researchers, capable of running the benchmark, writing scripts, debugging failures, and improving a local policy, yet unable to operationalize abstract data-research ideas without additional structure.

Table 17: Interventions with light-scaffolding and observed failure modes.

Intervention	Intended effect	Observed behavior
Standard instruction	Let the agent autonomously improve data.	Defaults to filtering, deduplication, simple score thresholds.
Ask for broad solutions	Encourage movement across policy families.	Plans mention many families; implementation remains local.
Ask to avoid heuristics	Suppress trivial length/score filters.	Agent renames heuristics as “quality criteria” or uses them indirectly.
Provide flat skill list	Expose many approaches.	Skills are rarely used or used superficially without correct evidence.
Ask to inspect raw eval logs	Encourage targeted rewriting.	Agent reads summary metrics or selects examples, but rarely rewrites with grounded error taxonomy.

C.3 Heavy Scaffolds

I. Self-Research reshapes the workflow into a structured research loop where every experiment must be motivated by a concrete observation, documented with a hypothesis, and logged with its findings–positive or negative–to inform the next iteration. Examples of agent-written research plans are provided in the code repo.

Heavy-scaffolding I: Self-research

- *Research-first mandate* requiring that every iteration (beyond baselines) begin with actual data research rather than intuitive code changes. Before implementing the curation script, the agent is instructed to first write a research.md for the iteration containing an observation, a hypothesis, and a minimal change — all grounded in concrete data findings, not generic intuition.
- *Blocklist of exhausted strategies:* The research version adds an explicit list of patterns that are "out" — strategies based on trivial operations and shallow ideas.
- *Richer result logging schema:* The agent is instructed to document outcomes at each iteration with an explicit template including three additional fields–"skill ref" (which prior research.md commit this iteration builds on), "failure mode" (what new finding the run revealed, tied to specific benchmarks), and "next skill candidate" (the next research question to investigate). This creates a chain of reasoning across iterations to discourage ungrounded moves without a reasonable rationale or logic.

II. Adapt Research Papers hardens the soft "prioritize skills" guidance from light-scaffolding II into a strict protocol: every iteration must cite a specific skill, adapt a concrete procedure from it, rotate across skill categories, and avoid a defined set of low-effort heuristics. The result is a more disciplined and auditable research loop.

Heavy-scaffolding II: Adapt Research Papers

- *Session initialization*: listing skills -> deeply reading skills: at the start of the session, the agent must browse all skill categories, pick at least 3 skills per category that seem relevant, and read those skill cards in full before writing its curation script. The agent is also prompted to use 'Agent'/'Explore' tools to scan quickly through the one-line decision notes in the skill cards. The pipeline raises the bar from awareness to comprehension.
- *Strategy must be skill-grounded*: formalizing the requirement that the curation script at each iteration must be grounded in a specific skill card. The workflow is: identify a failure mode from the last result, pick 1–3 relevant skills, read them fully, then state in the logs a specific skill card and a one-line adaptation. This changes implicit/soft guidance in the light-scaffolding ("prioritize exploration with reference skills") to a hard rule.
- *Category rotation requirement*: the heavy-scaffolding requires that before repeating a skill category, the agent must attempt at least one iteration from each of the other three. The 'next skill candidate' must also come from a category not used in the last three iterations unless justified.
- *Blocklist of exhausted strategies* and *Richer result logging schema* are enforced similarly to Heavy-scaffolding I. The agent is forbidden from citing a skill as mere "inspiration" rather than an adapted procedure. Not citing a specific skill card in a non-baseline row is explicitly called a "rule violation."

C.4 Rewriting Scaffolds

Scaffolding for rewriting. The rewriting scaffold introduces two capabilities absent from the selection-only scaffolds. First, *eval-log reading*: after each evaluation, the agent samples 5–15 incorrect predictions per benchmark, categorizes the failure mode (e.g., OCR misread, format violation, hallucinated option, missing reasoning step), and cross-references the training data to identify which examples–or gaps–produced the failure. Second, *targeted rewriting*: the agent constructs a rewrite recipe specifying (a) which slice of the training data to target, (b) what transformation to apply via the MLLM, and (c) the expected effect on the identified failure mode. The recipe is logged alongside the iteration result, creating an auditable chain from failure observation to rewrite action to training outcome. Full scaffold details are in Appendix G.

We adopt the self-research heavy scaffold for rewriting. Each iteration must begin with evidence from evaluation logs or data inspection, name a concrete failure mode, select source examples tied to that failure, rewrite them with an external MLLM tool, and validate the rewritten outputs before training. The scaffold therefore changes the task from "pick the best examples" to "pick and improve examples under a fixed training budget." Compared with template rewriting, the setup is more agentic: the rewrite policy is not fixed in advance, and the agent must decide which examples to rewrite, what transformation to apply, and how aggressively to use the rewriting model.

D Experimental Details and Additional Results

D.1 Terminal Agents

All agents interact with the CURATION-BENCH environment through a terminal interface: they read files, write and edit scripts, run shell commands, and observe stdout/stderr. The agents differ in their backbone model, harness, and hosting, but share the same workspace, CLI, and submission gate.

Claude Code. We use the Claude Code terminal agent (Anthropic, 2025) with two backbone models: Opus 4.7 (primary) and Sonnet 4.6 (ablation). Claude Code provides built-in tool use for file editing, command execution, and multi-step planning. No additional plugins or custom tools are added beyond the benchmark CLI.

Codex. We use OpenAI's Codex terminal agent with two backbone models: GPT-5.4 (primary) and GPT-5.3 (ablation). Codex operates in a sandboxed environment with the same file-system layout and CLI access as the Claude Code runs.

Open-source models via OpenHands. To evaluate open-source agent backbones, we use the OpenHands harness (Wang et al., 2025), which wraps an LLM in a terminal-agent loop with file editing, shell execution, and observation parsing. We evaluate two models: Kimi K2.5 and Qwen3.5-397B. Both are served via the Together AI API[2] using an OpenAI-compatible chat-completion endpoint, which OpenHands consumes as its backend LLM. Inference parameters (temperature, max tokens) follow the OpenHands defaults for each model.

Shared configuration. Across all agents, the Docker workspace, candidate-pool access, benchmark CLI, submission validation, contamination audit, training recipe, and evaluation suite are identical. The only differences are the agent backbone and harness; no agent receives task-specific tools or data-curation expertise beyond what the scaffold condition specifies (Table 4).

D.2 DataComp Filtering Setup

For the CLIP pretraining instantiation, all agent and baseline submissions use the DataComp-Small filtering-track configuration described in Section B: the same 12.8M-pair candidate pool (of which 8.77M pairs were recoverable after link rot, see Section B), the same OpenCLIP ViT-B/32 training recipe (Table 14), and the same 38-task zero-shot evaluation suite (Gadre et al., 2023). This subsection records only the differences in *what was run* rather than *how it was run*.

Agent We evaluate Claude Code (Opus 4.7) on this instantiation. The agent operates over the metadata fields available in the DataComp pool, implements a filtering policy, submits the filtered subset through the harness CLI, and revises after observing the DataComp-suite score.

Baseline We compare against the published *top*-30% *CLIP score* DataComp filtering baseline (Gadre et al., 2023): pairs are kept when their similarity score under a frozen CLIP ViT-L/14 reference encoder is in the top 30% of the pool, after which a ViT-B/32 is trained from scratch under the same Small-track recipe.

Resources per session A single training-and-evaluation cycle for one filtered subset takes $\sim$4 hours of H100 time ($\sim$2 h training, $\sim$2 h evaluation, Table 14).

D.3 Shared Training and Evaluation Protocol

In experiments with fine-tuning pre-trained LLaVA-1.5-7B models on examples from the LLaVA-665K dataset, all submitted datasets are trained and evaluated through the same wrapper. Each submitted subset is fine-tuned for one epoch with the fixed recipe in Table 10 and evaluated by the same eight-benchmark VLMEvalKit pipeline (Duan et al., 2024).

[2] https://www.together.ai/

Source data and model. All subsets are drawn from the LLaVA-665K mixture (Liu et al., 2024a) released on Hugging Face at https://huggingface.co/datasets/liuhaotian/LLaVA-Instruct-150K/blob/main/llava_v1_5_mix665k.json, materialized locally as five Arrow splits: coco (364,100), gqa (72,140), ocr_vqa (80,000), textvqa (21,953), vg (86,417), and text_only (40,688). Training always starts from the same pre-data-curation LLaVA-1.5-7B checkpoint (Liu et al., 2024a), released with the artifact as HF checkpoint.

Training. The per-model procedure is the same across baseline families, budgets, and agent submissions. Each submitted dataset is fine-tuned on one NVIDIA A100 GPU from the same LLaVA-1.5-7B initialization for one epoch with the vision tower frozen. Training uses PyTorch (Paszke et al., 2019) with Accelerate support (Gugger et al., 2022). The trainer uses the same LLaVA chat-template normalization and assistant-token masking for all subsets. Table 10 summarizes the fixed recipe.

Evaluation. Each trained checkpoint is evaluated through the repository's VLMEvalKit wrapper rather than by manually invoking benchmark scripts. The wrapper isolates each benchmark's artifacts, archives stale outputs before reruns, and refuses eval-only mode if the expected trained checkpoint is missing. All checkpoints are evaluated on the same eight VLMEvalKit benchmarks: MMVet (Yu et al., 2023), LLaVA-Bench (Liu et al., 2023a), MMBench (Liu et al., 2024b), MMMU validation (Yue et al., 2024), MMStar (Chen et al., 2024), MathVista-Mini (Lu et al., 2023), OCRBench (Liu et al., 2024c), and HallusionBench (Guan et al., 2024). Candidate-model inference is run by VLMEvalKit (Duan et al., 2024) with vLLM (Kwon et al., 2023). Judge-based metrics use a separate OpenAI-compatible Qwen3.5-27B endpoint (Team, 2026) for all baseline runs. The server-side judge configuration is summarized in Table 18.

Table 18: Judge-server configuration for judge-based VLMEvalKit metrics. This endpoint is separate from candidate-model evaluation inference.

Setting	Value
GPU type	2 NVIDIA RTX A6000 GPUs
Model	Qwen/Qwen3.5-27B
Tensor parallel size	2
GPU memory utilization	0.90
Maximum model length	2048 tokens
Maximum concurrent sequences	64
Precision	bfloat16
Reasoning parser	qwen3
Thinking mode	Disabled via enable_thinking=false
Request defaults	VLMEvalKit/OpenAI-wrapper defaults unless a benchmark wrapper overrides them: temperature=0, max_tokens=2048, n=1.

Aggregation and reporting. For each trained model, the aggregate score is the mean of the eight benchmark values after applying the transformations in Table 19. These transformations put all components on an approximate 0–100, higher-is-better scale. The resulting average is the run-level score used for reporting.

D.4 Non-Agent Baseline Construction

This subsection documents the non-agent baselines for fine-tuning pre-trained LLaVA-1.5-7B models on examples from the LLaVA-665K dataset: simple sampling references, human-designed selection baselines, and a rewriting baseline. All baselines are trained and evaluated with the shared protocol in Section D.3.

Baseline families. Table 20 summarizes the non-agent references. ICONS and ARDS are used as released selected-example pools: we randomly sample the requested budget from each pool. Template

Table 19: Benchmark normalization used to compute the aggregate score.

Benchmark	Raw value	Value used in aggregate
MMVet	Score reported by VLMEvalKit	Unchanged
LLaVA-Bench	Score reported by VLMEvalKit	Unchanged
MMBench	Accuracy in $[0, 1]$	$100\times$ accuracy
MMMU validation	Accuracy in $[0, 1]$	$100\times$ accuracy
MMStar	Accuracy in $[0, 1]$	$100\times$ accuracy
MathVista-Mini	Score reported by VLMEvalKit	Unchanged
OCRBench	Score out of 1000	Score $/10$
HallusionBench	aAcc, fAcc, and qAcc	Mean of aAcc, fAcc, and qAcc

Matters first samples from LLaVA-665K dataset and then applies a programmatic template rewrite; this baseline does not call an LLM.

Table 20: Non-agent baseline families.

Baseline	Construction	Reference / artifact
Random	Sample B examples from LLaVA-665K	HF data
ICONS	Sample B examples from the released ICONS pool	(Wu et al., 2024); GitHub
ARDS	Sample B examples from the released ARDS pool	(Yang et al., 2025); GitHub
Template Matters	Sample from LLaVA-665K, then apply template rewrite	(Wang et al., 2024b); GitHub

Budgets and replicates. Random and Template Matters are evaluated at 10k, 20k, 50k, 100k, and 200k examples. ICONS and ARDS are evaluated at 10k, 20k, 50k, and 100k examples. For each baseline family and subset size, we generate up to six independently seeded runs. Within each run, all baseline-family generators share the same run-level seed, so method comparisons are paired when the corresponding method-size cells are available. Different runs use different seeds. We report the mean and sample standard deviation across available run-level scores. If a method-size cell has fewer than six completed runs, we report the available count rather than treating the cell as complete.

D.5 Additional Results and Ablation Studies

We run four additional ablations along the axes of CURATION-BENCH's task specification and the agent setup. Each ablation primarily varies one axis while holding the rest fixed at the original open-prompt setting (LLaVA-665K $\rightarrow$ LLaVA-1.5-7B, $B = 10$k, default scaffold, Claude Code Opus 4.7 backbone). The budget and scaffold ablations additionally replicate across both Claude and Codex backbones to test whether their effects hold across model families.

Selection budget. We vary $B \in \{10\text{k}, 20\text{k}, 50\text{k}\}$ on fine-tuning pre-trained LLaVA-1.5-7B models on examples from the LLaVA-665K dataset curated by Claude Code (Opus 4.7)/Codex (GPT-5.4). This tests whether the agent's gain over a random subset scales with budget. **Task generalization.** We vary M_0 and $\mathcal{D}_{\text{pool}}$ at $B = 10$k under Claude Code Opus 4.7: two alternative target models—*Qwen2-VL-2B* (base-model fine-tuning) and *Qwen2.5-VL-3B* (post-training)—and one alternative data pool, *Vision-Flan*. This tests whether the curation strategies the agent develops transfer across model scales, training regimes, and pool structures, or are tuned to LLaVA-665K's specific subset organization. **Agent backbone.** We compare four agent backbones at $B = 10$k under the default scaffold: *Claude Code with Opus 4.7*, *Claude Code with Sonnet 4.6*, *Codex with GPT-5.4*, and *Codex with GPT-5.3*. This tests how different agent backbones perform on the same task and scaffold. **Scaffold.** We compare three scaffolds at $B = 10$k under both Claude and Codex: *open-prompt* (minimal-scaffolded), *Heavy Scaffold I* (self-research), and *Heavy Scaffold II* (adapt papers).

Table 21 shows results for interventions with heavy-scaffolding (Codex). Both heavy scaffolding substantially increase grounded iterations while only Heavy Scaffold II (Adapt papers) significantly increases iterations with new-policy-family moves and eliminates shallow operations, which also leads to visibly improved outcomes.

Table 21: Interventions with heavy-scaffolding (Codex). Autonomous data curation experiments: each row is one commit produced by the agent. "Keep" rows improved the composite accuracy; "discard" rows were rolled back. Reporting avg. scores and "% of full gain" (% of the improvement from the base model to full-data fine-tuning). The four right-most columns rate each iteration on whether it introduces a new policy family, is grounded in evidence, is effective (improves accuracy), or is trivial/heuristic. ○ = yes, ✗ = no. Green numbers show improvement over the open-prompt baseline. Best accuracy and quality scores marked in **bold**

Intervention	Avg. score/%	Description	New Policy	Grounded	Effective	*Shallow*	Execution
Open-prompt (baseline, 10k)	*33.53 / 57%*	OCR text-rich ratio 60/40	*3/10*	*4/10*	***4/10***	*6/10*	*10/10*
Self-research (Heavy I, 10k)	32.96	soften region bonus in weighted sampler	3/10	**10/10 (+6/10)**	3/10	6/10	10/10
Adapt papers (Heavy II, 10k)	**34.28 / 66%** **(+0.75/+9%)**	ALIGN/RedCaps-style minimal filtering with community-supported noisy pool retention	**9/10 (+6/10)**	**10/10 (+6/10)**	3/10	**1/10 (-5/10)**	10/10
Random 100k	$33.7_{\pm 0.2}$/$59\%_{\pm 3\%}$	-	-	-	-	-	-
Random 200k	$34.0_{\pm 0.2}$/$63\%_{\pm 3\%}$	-	-	-	-	-	-
ICONS 100k (Wu et al., 2024)	34.5 / 69%	-	-	-	-	-	-
ARDS 100k (Yang et al., 2025)	34.1 / 64%	-	-	-	-	-	-

Table 22 shows results for ablation studies with **different agent backbone** on the multimodal instruction-tuning data-selection task. Different agent backbones moderately affect the final outcome without changing the general patterns in comparison with the random baseline or human solutions.

Table 22: Results for ablation studies with **different agent backbone** on the multimodal instruction-tuning data-selection task. Scores averaged over 8 benchmarks. Green numbers show improvement over the random 10k baseline. "% of full gain" column measures the fraction of the improvement from the base model to full-data fine-tuning that is achieved by each setting. Best outcomes are marked in **bold**. Different agent backbones moderately affect the final outcome without changing the general patterns in comparison with the random baseline or human solutions.

Setting	Data size	Avg. score	Δ over base	% of full gain	Notes
Base model	0	28.8	0.0	0%	pre-trained
Full-data fine-tuning	665k	37.1	+8.3	100%	official checkpoint
Random 10k (baseline)	*10k*	*32.5*	*+3.7*	*45%*	*Best of 10 runs*
ICONS 10k	10k	33.3	+4.5	54%	Wu et al. (2024)
ARDS 10k	10k	33.2	+4.3	52%	Yang et al. (2025)
Agent (open-prompt)	10k	**34.2 (+1.7)**	+5.3	**65% (+20%)**	Claude Code (Sonnet 4.6), 10 iter.
Agent (open-prompt)	10k	33.3 (+0.8)	+4.5	54% (+9%)	Codex (GPT-5.3), 10 iterations
Random 50k	50k	$33.4_{\pm 0.2}$	$+4.6_{\pm 0.2}$	$55\%_{\pm 3\%}$	Uniform Downsampling
Random 100k	100k	$33.7_{\pm 0.2}$	$+4.9_{\pm 0.2}$	$59\%_{\pm 3\%}$	Uniform Downsampling
ICONS 20k	20k	33.6	+4.8	58%	Wu et al. (2024)
ICONS 50k	50k	33.9	+5.1	61%	Wu et al. (2024)
ARDS 20k	20k	33.4	+4.6	55%	Yang et al. (2025)
ARDS 50k	50k	33.8	+5.0	60%	Yang et al. (2025)

Table 23 shows results for **open-prompt** generalist agents on selecting 20k/50k data from LLaVA 665k instruction-tuning dataset for training LLaVA-1.5-7B models. The open-prompt agents achieve general improvements over random baselines and reach or exceed evaluated human baselines across data sizes.

Table 24 shows results for ablation studies with **different data tasks (different models/datasets)** on multimodal instruction-tuning data-selection. The open-prompt agents achieve general improvements over random baselines across the tasks.

D.6 Example Agent Traces and Trajectory Annotations

Examples of trace annotations for autonomous data curation with Codex (open-prompt) are provided in Table 25.

Table 23: **Open-prompt** generalist agents on selecting 20k/50k data from LLaVA 665k instruction-tuning dataset for training LLaVA-1.5-7B models. Scores are averaged over 8 benchmarks; "% of full gain" is the fraction of the base→full-data fine-tuning improvement recovered. Green deltas are over the random baseline at the same budget. Best per budget in **bold**.

Setting	Avg. score	Δ over base	% of full gain
Base model (no fine-tuning)	28.8	0.0	0%
Full-data fine-tuning (665k)	37.1	+8.3	100%
20k budget			
Random (baseline)	33.0±0.2	+4.2	51%
ICONS (Wu et al., 2024)	33.6	+4.8	58%
ARDS (Yang et al., 2025)	33.4	+4.6	55%
Agent – Claude Code (Opus 4.7), 10 iter.	**33.8 (+0.8)**	+5.0	**60% (+9%)**
Agent – Codex (GPT-5.4), 10 iter.	33.7 (+0.7)	+4.9	59% (+8%)
50k budget			
Random (baseline)	33.4±0.2	+4.6	55%
ICONS (Wu et al., 2024)	33.9	+5.3	61%
ARDS (Yang et al., 2025)	33.8 (+0.4)	+5.0	60% (+5%)
Agent – Codex (GPT-5.4), 10 iter.	**34.0 (+0.6)**	+5.2	**63%(+8%)**
Random 100k (reference)	33.7±0.2	+4.9	59%

Table 24: Results for ablation studies with **different data tasks (different models/datasets)** on multimodal instruction-tuning data-selection. Scores averaged over 8 benchmarks. Green numbers show improvement over the random 10k baseline. Best outcomes are marked in **bold**. The open-prompt agent (with Claude Code) achieves general improvements over random baselines across the tasks.

Setting	Data size	Avg. score	Notes
Qwen2-VL-2B + LLaVA-665K dataset			
Random 10k (baseline)	*10k*	*48.1*	*Uniform Downsampling*
Agent (open-prompt)	10k	**48.5 (+0.4)**	Claude Code (Opus 4.7), 10 iter.
Qwen2.5-VL-3B + LLaVA-665K dataset			
Random 10k (baseline)	*10k*	*63.1*	*Uniform Downsampling*
Agent (open-prompt)	10k	**63.5 (+0.4)**	Claude Code (Opus 4.7), 10 iter.
LLaVA-1.5-7B + Vision-Flan dataset			
Random 10k (baseline)	*10k*	*31.8*	*Uniform Downsampling*
Agent (open-prompt)	10k	**32.2 (+0.4)**	Claude Code (Opus 4.7), 10 iter.

Examples of agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (open-prompt) are provided in Table 26.

Examples of agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Light Scaffold I, Data strategies) are provided in Table 27.

Examples of agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Light Scaffold II, Research papers) are provided in Table 28.

Examples of agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Heavy Scaffold I, Self-research) are provided in Table 29.

Examples of agent logs on "Failure Mode" and "Next Skill Candidate" at each iteration for autonomous data curation corresponding to the trajectory in Table 29 are provided in Table 30.

Examples of agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Heavy Scaffold II, Adapt papers) are provided in Table 31.

Examples of agent logs on "Failure Mode" and "Next Skill Candidate" at each iteration for autonomous data curation corresponding to the trajectory in Table 31 are provided in Table 32.

Table 25: Trace annotations for autonomous data curation with Codex (open-prompt): each row is one commit produced by the agent. "Keep" rows improved the composite accuracy; "discard" rows were rolled back. The four right-most columns rate each iteration on whether it introduces a new policy family, is grounded in evidence, is effective (improves accuracy), or is shallow/trivial. ○ = yes, ✗ = no. Best accuracy marked in **bold**.

Commit	Avg. score	Status	Description	New Policy	Grounded	Effective	*Shallow*
0000000	28.93	keep	Pre-init baseline (no finetune)	–	–	–	–
cf218fe	32.53	keep	Baseline random 10k	–	–	–	–
c8cb321	32.68	keep	Near-random visual, no text_only	○	○	○	✗
28806e1	33.24	keep	Stratified random reweight	○	○	○	✗
778e306	32.13	discard	COCO-rebalance tradeoff	✗	✗	✗	○
d5ac779	32.84	discard	Midpoint mix: stronger MMVet/LLaVA, weak MMBench/Math	✗	✗	✗	○
a19f32c	33.01	discard	Seed sweep: better MMBench/Math, weaker MMVet/OCR	✗	✗	✗	○
ebac17e	32.93	discard	Evenly-spaced within subset quotas	✗	✗	✗	○
bbdf052	33.27	keep	OCR/textvqa text-rich mixed sampling	○	○	○	✗
489dae9	32.40	discard	Seed sweep for OCR text-rich mix	✗	✗	✗	○
e4578e2	32.11	discard	Mixed seeds: main=123, OCR=3407	✗	✗	✗	○
b88a84f	**33.53**	keep	OCR text-rich ratio 60/40	✗	○	○	✗
Execution success rate: 10/10			**Progression on data:**	3/10	4/10	4/10	6/10

Table 26: Agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (open-prompt). Each row is one commit produced by the agent. "Keep" rows improved the composite accuracy; "discard" rows were rolled back.

Commit	Avg. score	MMVet	LLaVABench	OCRBench	HalluBench	MMMU	MathVista	MMStar	MMBench	Status	Description
0000000	28.95	17.8	26.0	240	26.8	31.80	25.7	33.8	45.60	keep	Pre-init baseline (no finetune)
9327618	32.38	26.7	37.7	291	29.2	31.78	24.1	30.8	49.74	keep	Baseline random 10k
d5bf0c3	33.06	28.0	37.3	290	30.0	34.11	23.3	33.6	49.14	keep	Balanced subset, OCR boost, no text_only
9490f21	32.88	28.1	41.5	300	27.1	33.78	22.2	32.5	47.85	discard	Balanced subsets + longest asst. responses
117adc1	32.06	28.1	36.9	285	30.3	28.11	24.0	32.7	47.93	discard	Balanced subsets + text_only (500)
8e9876a	33.23	29.8	37.4	288	32.5	30.67	23.8	34.6	48.28	keep	Equal allocation 2000 each visual subset
5647fa1	32.56	29.0	32.8	287	32.8	30.67	23.6	34.9	48.02	discard	Boost VG/GQA, reduce COCO/OCR
c6c0c38	33.63	28.6	39.0	294	30.5	34.89	23.3	33.7	49.66	keep	Near-equal: COCO=2500, others=1875
a32439c	32.95	29.9	34.4	288	30.4	34.56	23.5	33.2	48.93	discard	COCO=3000, others=1750 (too much COCO)
72cad7b	33.70	29.4	38.8	296	31.15	34.11	23.8	33.2	49.55	keep	OCR-boosted: COCO=2000, OCR/-Text=2250
85fc5a3	33.02	28.0	35.9	296	32.64	31.33	23.3	34.9	48.45	discard	heavy OCR+VG: COCO=1500, too little COCO hurts
2c09740	33.74	30.4	38.9	296	33.87	30.00	24.7	33.9	48.54	keep	GQA+textvqa boost: COCO=2000, GQA=2250, textvqa=2250

Table 27: Agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Light Scaffold I, Data strategies). Each row is one commit produced by the agent. "Keep" rows improved the composite accuracy; "discard" rows were rolled back.

Commit	Avg. score	MMVet	LLaVABench	OCRBench	HalluBench	MMMU	MathVista	MMStar	MMBench	Status	Description
0000000	28.84	17.9	25.7	238	26.7	31.40	25.9	33.8	45.50	keep	Pre-init baseline (no finetune)
c3ba68c	32.49	26.6	35.3	295	27.9	35.30	22.9	32.9	49.60	keep	Random 10k baseline
3d4d1d5	32.42	25.7	36.7	295	27.8	34.80	24.6	33.1	47.20	keep	Subset balanced (3k COCO / 2k OCR / 2k GQA / 1.5k VG / 1.5k TextVQA)
4295574	32.31	26.5	37.5	286	28.4	34.20	23.6	33.3	46.50	discard	quality-weighted sampling sqrt(asst_len) - hurt factual benchmarks
fcf3d46	33.14	25.5	38.7	289	30.0	35.10	24.0	33.4	49.40	keep	COCO task diversity + orig quotas + 15% floor (best in session)
c7b7c12	33.06	28.2	41.5	293	27.8	33.70	22.3	32.9	49.10	discard	soft quality pref (longest responses) - verbose COCO bias hurt
218f5bd	32.44	29.4	36.2	291	29.8	29.30	23.5	32.1	50.20	discard	OCR-boosted quotas - reducing GQA hurt MMMU badly (29.3)
fa7b881	33.06	25.6	40.2	292	28.8	35.40	22.9	32.4	49.80	discard	question-hash stratified sampling - no improvement
b2f9da3	32.42	27.2	36.7	297	27.1	35.40	22.7	30.4	50.10	discard	COCO tier-balanced (short/med/long) - no improvement
4ae030e	33.30	30.2	36.2	295	29.3	35.90	23.8	31.5	49.90	keep	8k samples proportional quotas (fewer=regularization?)
d5c210c	33.21	28.2	36.6	298	32.4	35.10	23.6	31.3	48.70	discard	6k samples - similar to 8k, slightly worse MMBench
9f08a09	33.97	29.7	40.3	303	29.5	35.60	23.6	33.0	49.80	keep	8k COCO-heavy (3k coco, 1k vg, 1.5k ocr, 1.5k gqa, 1k textvqa) - NEW BEST

Table 28: Agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Light Scaffold II, Research papers). Each row is one commit produced by the agent. "Keep" rows improved the composite accuracy; "discard" rows were rolled back.

Commit	Avg. score	MMVet	LLaVABench	OCRBench	HalluBench	MMMU	MathVista	MMStar	MMBench	Status	Description
0000000	28.95	17.8	26.0	240	26.8	31.80	25.7	33.8	45.60	keep	Pre-init baseline (no finetune)
f9b769f	32.19	26.6	35.9	292	28.9	32.60	24.0	30.3	50.10	keep	Baseline random 10k
72df18e	32.18	28.0	33.6	285	31.7	31.20	23.9	31.1	49.30	discard	balanced subset sampling with OCR boost
b8e2f93	32.09	28.5	38.7	292	26.9	31.60	23.4	31.4	47.00	discard	quality-based: longest responses per subset
3dd1aee	33.29	29.1	32.4	290	32.7	34.80	24.2	34.6	49.50	keep	minimal COCO no text_only boost visual diversity
d47b299	32.81	27.9	34.3	290	35.1	29.30	24.2	35.2	47.50	discard	near-zero COCO 500 boost VG 3k OCR 2.5k
199733f	32.77	29.0	34.0	286	33.2	33.40	23.3	32.3	48.40	discard	text_only 500 back, ocr_vqa 1500
55275b8	33.41	27.4	35.6	286	33.0	35.10	24.1	35.3	48.20	keep	7k samples same proportions
6565487	32.76	27.8	37.0	297	31.0	33.80	22.7	33.6	46.60	discard	7k quality-filtered remove shortest 25% per subset
5941975	33.72	30.1	37.9	283	32.6	35.30	24.2	33.5	47.80	keep	8k samples winning proportions scaled up
1e7b23b	32.67	28.3	37.3	279	33.9	26.70	24.2	34.1	48.90	discard	9k samples winning proportions
679ba47	32.20	29.7	33.8	279	33.5	26.70	23.7	34.5	47.90	discard	8k boost OCR/TextVQA reduce VG/GQA

Table 29: Agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Heavy Scaffold I, Self-research). Each row is one commit produced by the agent. "Keep" rows improved the composite accuracy; "discard" rows were rolled back.

Commit	Avg. score	MMVet	LLaVABench	OCRBench	HalluBench	MMMU	MathVista	MMStar	MMBench	Status	Description
0000000	28.95	17.8	26.0	240	26.8	31.80	25.7	33.8	45.60	keep	Pre-init baseline (no finetune)
f9b769f	32.19	26.6	35.9	292	28.9	32.60	24.0	30.3	50.10	keep	Baseline random 10k
8fcb60b	32.61	26.1	39.9	285	28.2	33.80	23.0	31.9	49.50	keep	drop text_only, stratified visual 10k
b3f1382	33.12	26.4	39.0	299	29.1	34.70	23.9	31.8	50.30	keep	drop text_only+vg, stratified 4-subset 10k
0bf0abb	32.51	27.7	36.4	290	28.5	34.80	23.6	30.9	49.20	discard	2x OCR weight oversampling
58a5268	31.47	24.8	39.3	290	27.1	26.00	24.4	32.3	49.00	discard	multi-turn coco priority (all 3+ turns)
424754d	32.59	25.7	37.1	295	27.5	34.60	23.7	32.3	50.30	discard	reduce GQA to 500, extra to coco
ead8bac	32.32	24.4	37.1	293	26.5	35.20	24.7	31.1	50.20	discard	proportional SEED=7 (variance test)
0dc4697	32.65	27.2	39.9	288	28.0	34.10	23.0	32.4	47.80	discard	COCO short-bias reduction (25% short / 75% non-short)
933dce7	32.24	26.9	40.4	287	27.4	28.70	22.9	32.1	50.80	discard	OCR rebalance (ocr_vqa-400, textvqa+400)
5da628e	31.75	28.2	39.0	288	26.6	26.00	22.8	31.9	50.70	discard	GQA diversity filter (4+ unique question types)
d2755be	33.32	26.3	41.8	289	26.8	35.60	23.7	31.9	51.60	keep	ocr_vqa-200 to coco for visual diversity

Table 30: Agent logs on "Failure Mode" and "Next Skill Candidate" at each iteration for autonomous data curation corresponding to the trajectory in Table 29. Each row is one commit produced by the agent. “Keep” rows improved the composite accuracy; “discard” rows were rolled back.

Commit	Avg. score	Status	Description	Failure Mode	Next Skill Candidate
0000000	28.95	keep	Pre-init baseline (no finetune)	weak on OCR and math; hallucination issues	none (random baseline to run)
f9b769f	32.19	keep	Baseline random 10k	weak math/logical reasoning; OCR improved over pre-init	(first skill to explore)
8fcb60b	32.61	keep	drop text_only, stratified visual 10k	LLaVABench +4.0 but OCR regressed −7; removing text_only helped conversation/detail but lost some OCR signal; gqa short answers may be diluting useful signal	which coco samples resemble OCRBench/-MathVista test items?
b3f1382	33.12	keep	drop text_only+vg, stratified 4-subset 10k	OCR recovered vs iter1; MMMU +1.9 over baseline; MathVista still weak at 23.9; gqa short-answer training may not transfer to complex reasoning	can we improve detailed reasoning by biasing coco towards longer, multi-step responses?
0bf0abb	32.51	discard	2x OCR weight oversampling	OCR oversampling didn’t improve OCR-Bench (290 vs 299); reduced coco budget hurt LLaVABench/MMBench; coco diversity > OCR-specific volume	within coco, do detailed multi-turn conversations transfer better than short-answer samples?
58a5268	31.47	discard	multi-turn coco priority (all 3+ turns)	MMMU crashed 34.7→26.0; removing 1-turn MC samples destroys MC-format performance; diversity within coco matters more than multi-turn bias	what makes iter2’s proportional split work well? try different seeds to measure selection variance
424754d	32.59	discard	reduce GQA to 500, extra to coco	HallusionBench dropped −1.6; GQA yes/no format helps hallucination discrimination more than expected; proportional balance is hard to beat	can we improve by keeping proportional but using diversity sampling within subsets?
ead8bac	32.32	discard	proportional SEED=7 (variance test)	seed variance confirmed: −0.008 vs SEED=42; MMMU/MathVista slightly better but MMVet/LLaVA/Hall worse; seed affects benchmarks differently	try SEED=100 to test whether higher seeds produce better draws
0dc4697	32.65	discard	COCO short-bias reduction (25% short / 75% non-short)	short COCO samples help OCR/MMBench more than hurt reasoning; OCRBench −11 and MMBench −2.5; MathVista unchanged	do OCR_VQA and short COCO samples share training signal? try removing redundant ones
933dce7	32.24	discard	OCR rebalance (ocr_vqa−400, textvqa+400)	OCR_VQA yes/no+factual format helps MMMU MC format (28.7 vs 34.7); TextVQA captioning does not help OCRBench	why does OCR_VQA help MMMU? is it the yes/no MC-format training?
5da628e	31.75	discard	GQA diversity filter (4+ unique question types)	MMMU needs repetitive yes/no format, not diverse Q types; diversity dilutes MC-format signal; high-diversity GQA has more turns which changes training dynamics	can image-level dedup within COCO improve visual diversity without changing format?
d2755be	33.32	keep	ocr_vqa−200 to coco for visual diversity	OCRBench −10 as expected; HallusionBench −2.3 unexpected; confirms MMMU crash was from TextVQA not OCR_VQA; COCO diversity helps LLaVA/MMBench	can we recover HallusionBench by tweaking GQA or adding hallucination-specific COCO samples?

Table 31: Agent trajectory and the corresponding task scores at each iteration for autonomous data curation with Claude Code (Heavy Scaffold II, Adapt Papers). Each row is one commit produced by the agent. “Keep” rows improved the composite accuracy; “discard” rows were rolled back.

Commit	Avg. score	MMVet	LLaVABench	OCRBench	HalluBench	MMMU	MathVista	MMStar	MMBench	Status	Description
0000000	28.95	17.8	26.0	240	26.8	31.80	25.7	33.8	45.60	keep	Pre-init baseline (no finetune)
f9b769f	32.19	26.6	35.9	292	28.9	32.60	24.0	30.3	50.10	keep	Baseline random 10k
6fe2c47	31.88	27.0	37.6	284	28.9	30.40	23.7	30.9	48.10	discard	D4 hash-dedup + BoW k-means diversification
03b05c0	31.93	29.6	38.2	290	29.1	28.70	22.8	32.1	46.00	discard	DEITA LLM-judge quality+complexity scoring on 30k pool
2505edd	31.53	25.7	36.7	284	27.2	29.30	24.1	31.5	49.30	discard	Self-Instruct MinHash dedup (Jaccard>0.7) + random sample
692fab3	32.02	27.1	36.0	287	29.1	27.30	24.6	32.3	51.10	discard	Cambrian visual-centric: images only + exact dedup + random
58a6fab	31.49	27.0	38.5	282	29.4	24.70	24.8	30.1	49.30	discard	SemDeDup TF-IDF question clustering, random per cluster
9602a05	33.71	27.7	32.7	279	31.1	36.40	25.1	32.6	56.20	keep	EL2N proxy: base model forward-pass loss, top-10k highest-loss from 50k pool
6177e53	31.99	27.1	37.9	285	27.6	30.80	23.9	31.3	48.80	discard	Moderate-difficulty band (p25–p90): random 10k from loss band
9dc1d88	33.70	26.1	35.2	299	31.3	35.10	25.2	33.1	53.80	discard	D4 diversification on EL2N top-20k: TF-IDF k-means 20 clusters, 500 highest-loss per cluster
4694fbf	34.86	29.1	40.4	290	29.2	36.30	25.7	32.1	57.10	keep	EL2N top-loss with p95 assistant-loss noise filter
7bb6675	33.72	24.5	37.9	284	30.0	33.20	23.7	34.1	57.80	discard	Coreset: 100k pool, p95 noise filter, TF-IDF k-means 20 clusters, top-loss per cluster

Table 32: Agent logs on "Failure Mode" and "Next Skill Candidate" at each iteration for autonomous data curation corresponding to the trajectory in Table 31. Each row is one commit produced by the agent. “Keep” rows improved the composite accuracy; “discard” rows were rolled back.

Commit	Avg. score	Status	Description	Failure Mode	Next Skill Candidate
`0000000`	28.95	keep	Pre-init baseline (no finetune)	weak on OCR and math; hallucination issues	none (random baseline to run)
`f9b769f`	32.19	keep	Baseline random 10k	weak math/logical reasoning; OCR improved over pre-init	(first skill to explore)
`6fe2c47`	31.88	discard	D4 hash-dedup + BoW k-means diversification	BoW text clustering too crude for semantic diversity; OCR and MMMU regressed	DEITA: proxy quality+complexity scoring
`03b05c0`	31.93	discard	DEITA LLM-judge quality+complexity scoring on 30k pool	quality scoring biased toward verbose responses; MMMU and MMBench regressed (-3.9/-4.1)	Beyond Neural Scaling Laws: moderate-difficulty selection
`2505edd`	31.53	discard	Self-Instruct MinHash dedup (Jaccard>0.7) + random sample	aggressive dedup removed 20% of data including valid question variations; hurt OCR/MMMU/Hallusion	Cambrian: vision-centric data balance
`692fab3`	32.02	discard	Cambrian visual-centric: images only + exact dedup + random	improved 5/8 benchmarks (MMStar+2.0, MMBench+1.0) but MMMU regressed -5.3 (lost text_only language transfer)	SemDeDup: TF-IDF question-based clustering + Cambrian visual-only
`58a6fab`	31.49	discard	SemDeDup TF-IDF question clustering, random per cluster	equal-per-cluster sampling destroyed natural distribution; MMMU -7.9 (academic content severely undersampled)	EL2N: base model forward-pass loss as data quality proxy
`9602a05`	33.71	keep	EL2N proxy: base model forward-pass loss, top-10k highest-loss from 50k pool	LLaVABench and OCR regressed (-3.2/-13) suggesting highest-loss includes noisy/corrupt samples that hurt generation quality	Dataset Cartography: filter ambiguous vs hard-but-wrong to remove noisy high-loss samples
`6177e53`	31.99	discard	Moderate-difficulty band (p25–p90): random 10k from loss band	moderate band is too similar to random; lost reasoning signal (MMMU -5.6, MMBench -7.4) from high-loss samples while LLaVABench recovered (+5.2)	LIMA: fewer but higher-quality; EL2N top-5k to test if concentration beats dilution
`9dc1d88`	33.70	discard	D4 diversification on EL2N top-20k: TF-IDF k-means 20 clusters, 500 highest-loss per cluster	near-matched EL2N (0.3370 vs 0.3371); OCR recovered (+20) and LLaVABench (+2.5) but MMVet/MMBench regressed slightly	ShareGPT4V: response quality scoring; decompose loss into user/assistant tokens
`4694fbf`	34.86	keep	EL2N top-loss with p95 assistant-loss noise filter	HallusionBench slightly regressed (-1.9) suggesting some filtered samples were actually hallucination-correcting examples	Active Learning: coreset selection on EL2N-scored pool for maximum diversity coverage
`7bb6675`	33.72	discard	Coreset: 100k pool, p95 noise filter, TF-IDF k-means 20 clusters, top-loss per cluster	equal per-cluster allocation (500/cluster) forced low-quality samples from small clusters; MMVet -4.6, MMMU -3.1 vs best	TRAK: scale up pool to 100k with pure p95 filter (no diversity)

E Extended Discussion

E.1 From autonomous execution to agent-assisted research

Our results support a sharp distinction between two modes of agent progress. In **autonomous execution**, the human supplies a relatively concrete objective and the agent performs steps with clear local verification (fixing code, running training, producing a valid submission). In **agent-assisted research**, the objective is underspecified, the policy space is discontinuous, and evaluation is sparse and noisy. Training-data curation belongs to the second category: there is no oracle that tells the agent whether a data strategy is scientifically meaningful before an expensive training run.

This distinction reframes the research agenda. Execution tasks may continue to benefit primarily from stronger models, better tools, and longer contexts. Data research additionally requires *better scaffolds*–representations of the data-policy space, mechanisms for grounding hypotheses, protocols for comparing policies under noise, and interfaces that turn prior research into executable agent actions.

E.2 Scaffolds as research artifacts

A scaffold is not merely a prompt; it is an experimental-design object that determines what kinds of reasoning and action the agent can perform. In our experiments, a weak scaffold produces plans while a strong scaffold produces valid data policies–so the same backbone can look like a different system depending on its scaffold. Future papers should therefore report scaffold details with the same rigor as model details: a score obtained under a detailed pipeline with examples is not directly comparable to a score obtained under an open-ended instruction.

E.3 The autonomy–performance trade-off

Heavy scaffolds can improve performance while reducing autonomy, and this is a feature to measure rather than a flaw to avoid. A fully open task asks whether the agent can *discover* the method; a heavily scaffolded task asks whether it can *execute and adapt* the method. Both are useful, but they answer different questions, and the field will be better served by reporting a scaffold-strength ladder–open instruction, broad prompt, skill list, executable skill card, scripted pipeline–rather than a single number.

F Broader Impacts and Considerations

F.1 Broader Societal Impacts

CURATION-BENCH and the accompanying findings sit at the intersection of two trajectories with mixed societal implications: the increasing capability of generalist coding agents and the increasing centrality of training-data curation to model behavior. We discuss the most direct positive and negative impacts we anticipate.

Potential positive impacts. First, automating training-data curation can substantially reduce the human labor and compute required to build effective models. Our strongest observed 10k subset recovers up to 71% of the full-data fine-tuning gain on 665k examples, suggesting that agent-curated subsets can lower the barrier to fine-tuning for practitioners with limited compute, including academic laboratories, public-sector users, and groups in low-resource settings who cannot afford full-pool training. Second, the framework makes data decisions more legible: trajectory diagnostics record *why* a given dataset was selected, which supports auditability of model training pipelines and is useful for downstream accountability. Third, by open-sourcing CURATION-BENCH and the scaffold ladder, we aim to make agent-driven data research a shared public artifact rather than a proprietary capability concentrated in a few labs.

Potential negative impacts. The same capabilities admit several concerning failure modes. *(i) Reduced human oversight of training data.* If agent-curated subsets are deployed without review, harmful biases, unsafe content, or underrepresented groups in the candidate pool may be silently amplified or removed by opaque scoring functions. *(ii) Acceleration of capability development for misuse.* A general recipe for data optimization can in principle be applied to data pools whose downstream models are designed for

harmful tasks (e.g., disinformation, targeted persuasion, surveillance-related vision tasks). The scaffolds we study–particularly the paper-adaptation scaffold that turns published methods into executable agent actions–reduce the engineering friction for any party attempting such optimization, not only beneficial ones. *(iii) Concentration of advantage.* The strongest results we report require capable backbone agents and non-trivial compute for repeated training/evaluation. If only well-resourced actors can run the heavy-scaffold loop effectively, agentic data curation could widen rather than narrow the gap between large industrial labs and the broader community.

F.2 Considerations and Safeguards

Mitigations we recommend. We encourage three concrete practices when building on this work. First, treat agent-curated datasets as candidate artifacts subject to human or programmatic review for representation, safety, and license compliance, rather than as final training inputs. Second, report scaffold strength and trajectory-level diagnostics alongside outcome scores, so that improvements can be attributed to the agent's actual research process rather than to opaque tuning. Third, retain contamination checks and held-out evaluations when operating agents that can read evaluation logs; CURATION-BENCH ships such checks and we recommend they remain enabled in derivative pipelines.

F.3 Released Artifacts and Residual Risk

The artifacts we release are a benchmark harness, scaffolds (prompt specifications), trace logs, and curated dataset *manifests* (i.e., indices into public datasets). We do not release new pretrained models or new raw multimodal datasets. We have nonetheless taken several precautions.

Source data. All candidate pools are existing public datasets–LLaVA-665K (Liu et al., 2024a) and Vision-Flan (Xu et al., 2024)–used under their original licenses. Manifests select subsets of these pools rather than redistributing images or text, which preserves the upstream license posture and avoids creating a new venue for problematic content.

Traceability and legibility. Every iteration's curation script, manifest, audit result, training output, evaluation scores, and run notes are persisted under a commit hash (P4 in Section 2). This supports post-hoc inspection of agent behavior and makes it possible to identify, retract, or correct individual iterations if downstream issues are discovered.

Scope of the released artifact. We judge the misuse risk of releasing CURATION-BENCH itself to be modest: it does not provide novel offensive capabilities, it operates on already-public datasets, and its primary contribution is methodological–an evaluation framework and scaffolds for studying agent behavior. We will release the benchmark with documentation that explicitly describes the contamination audit, the assumption that downstream users will retain it, and the recommendation that agent-curated datasets be reviewed before deployment.

F.4 Declaration of LLM Use

LLMs are central components of this work and are declared accordingly.

LLMs as the object of study. The agents we evaluate–Claude Code with Opus 4.7 and Sonnet 4.6, and Codex with GPT-5.4 and GPT-5.3–are themselves LLM-based systems, and they are the experimental subject rather than auxiliary tools. All claims about agent behavior, exploration, and scaffold sensitivity are claims about these systems running under the protocols described in Sections 3,4.

LLMs as tools inside the pipeline. Three additional LLM-driven components are part of the methodology rather than the writing process. *(i) Trace annotation.* The four trajectories labels in Table 1 (new policy, grounded, effective, shallow) are produced with LLM assistance using Claude Opus 4.7, following the rubric in Section 2.2. We treat these labels as diagnostic annotations rather than ground-truth scientific claims, and we provide rubrics and trace examples to support auditing. *(ii) Skill-card compilation.* The paper-derived skill library used by light scaffold II and heavy scaffold II is generated by an automated Claude Code pipeline against a corpus of data-curation papers, following the template in Appendix G. *(iii)*

Rewriting tool and judge. The rewriting experiments call Qwen3.5-2B/9B and Gemma3-4B as rewriting backbones, and judge-based VLMEvalKit metrics use a separate Qwen3.5-27B endpoint (Table 18). All of these are documented with prompts, model versions, and configurations sufficient for reproduction.

LLMs for writing assistance. Beyond the components above, LLMs were used only in standard writing-support roles (grammar, phrasing, and LaTeX formatting suggestions). All scientific claims, experimental designs, analyses, and conclusions are the authors'. This use of LLMs does not affect the methodology, results, or originality of the work, and per the LLM policy is reported here for transparency rather than as a methodological dependency.

G Prompts and Skill-Card Templates

Additional full prompts and example skill cards are provided in the code repository.

Self-Research Iteration: Shift 200 OCR_VQA to COCO

Research: Shift 200 OCR_VQA to COCO (not TextVQA)

Observation

Three prior experiments changed training data composition. All crashed MMMU:
– 933dce7 (ocr_vqa −400, textvqa +400): MMMU 34.7 → 26.0
– 5da628e (GQA diversity filter): MMMU 34.7 → 26.0
– 0dc4697 (COCO short-bias): MMMU 34.7 → 34.1 (minor)

The 933dce7 crash was attributed to OCR_VQA reduction, but it also doubled TextVQA. TextVQA is 100% captioning format ("Provide a one-sentence caption"), which is radically different from MMMU's MC format. The MMMU crash may have been from TextVQA's captioning format conflicting with MC-format capability, not from OCR_VQA reduction.

Supporting evidence: 0bf0abb increased OCR weight (oversampled OCR_VQA) and MMMU was 34.8 – essentially unchanged. This suggests OCR_VQA count changes don't affect MMMU much, and 933dce7's crash was from TextVQA doubling.

OCR_VQA content: 39% yes/no, 21% "Who wrote this book?", 21% "What is the title?" – highly formulaic book metadata extraction. Of the 4 retained subsets, OCR_VQA is the most formulaic and least diverse.

Hypothesis
Shifting 200 samples from OCR_VQA to COCO (NOT TextVQA) will:
1. Keep MMMU stable (no TextVQA change, OCR_VQA change is small)
2. Improve visual diversity (200 more unique COCO images)
3. Potentially improve MMVet/LLaVABench (+0.3–0.5 from more detailed COCO content)
4. Minor OCRBench impact (−3 to −5 from slightly less OCR training)

Expected net: +0.001 to +0.003 accuracy

Minimal Change

After computing proportional allocations:
– ocr_vqa: 1486 → 1286 (−200)
– coco: 6765 → 6965 (+200)
– gqa: 1340 (unchanged)
– textvqa: 408 (unchanged)

Instructions for automated compilation of skill cards compiled from research papers

Characteristics of the skills

– Four top-level buckets, each a stage of the VLM data pipeline: data-acquisition, data-curation, data-selection, data-synthesis
– Each skill is one folder containing exactly one file: 'SKILL.md' with a title that combines the paper's short title and source.
– 1 paper -> 1 skill. No multi-paper aggregations; no extra assets (no code, no figures, no PDFs).

Per-file structure (rigidly templated)

Every ‘SKILL.md‘ uses the same H2 sections in the same order:

1. ‘# {Paper title}‘
2. **One-line decision** — when to use / when not to (single sentence each).
3. **Skill metadata** — ‘skill type‘, ‘paper kind‘ (e.g. ‘operational-method‘), ‘actionability‘ (high/med/low), ‘evidence quality‘ (‘full_paper‘).
4. **Goal** — one paragraph operational restatement.
5. **Problem signature** — modality, data state, scale regime, model requirement.
6. **Use when** / **Do not use when** — bullet pairs.
7. **Required inputs** / **Optional inputs** / **Outputs** — bolded named slots with one-line descriptions.
8. **Assumptions and prerequisites**.
9. **Procedure** — numbered steps, each with ‘Action: … / Why: … / Note: See paper for details.‘
10. **Parameters to set** — Role / How to set / Default-or-range / Effect.
11. **Validation checks**, **Failure modes**.
12. **Adaptation notes for VLM training** — always present, ties the paper back to VLM data work even when the source paper is not VLM-specific.
13. **Implementation notes**.
14. **Evidence from the paper** — 3–5 bullets with the load-bearing claims.
15. **Source paper** — title, year, venue, paper ID, URL, arXiv ID.

Tone / philosophy

– Decision-first ("Use this skill when…" / "Avoid it when…") rather than paper summary.

– Procedures are skeletons — the *shape* of the method, not full reproduction. Each step ends with "See paper for details."

– Knowledge-cards optimized for a downstream agent to retrieve and act on, not for human reading. They look like cookbook recipe cards keyed by paper.

Skill cards compiled from research papers on *data curation vision-language models (VLMs)*

Characteristics of the skills

– Four top-level buckets, each a stage of the VLM data pipeline: data-acquisition, data-curation, data-selection, data-synthesis
– Each skill is one folder containing exactly one file: ‘SKILL.md‘ with a title that combines the paper’s short title and source.
– 1 paper -> 1 skill. No multi-paper aggregations; no extra assets (no code, no figures, no PDFs).

Per-file structure (rigidly templated)

Tone / philosophy

– Decision-first ("Use this skill when…" / "Avoid it when…") rather than paper summary.

– Procedures are skeletons — the *shape* of the method, not full reproduction. Each step ends with "See paper for details."

– Knowledge-cards optimized for a downstream agent to retrieve and act on, not for human reading. They look like cookbook recipe cards keyed by paper.

Light-scaffolding I: General Data Strategists

Curation strategies to explore

Ideas (not limited to these):

– **Subset balancing**: Analyze source_subset distribution and adjust proportions.
– **Quality filtering**: Score by text quality (length, complexity, vocabulary diversity).
– **Diversity sampling**: Cover diverse image types and question categories.
– **Task-type targeting**: Prioritize samples matching eval benchmarks (OCR, visual reasoning, open-ended QA).
– **Deduplication**: Remove near-duplicates to make room for diverse samples.
– **Difficulty scoring**: Balanced mix of easy/medium/hard.
– **Conversation structure**: Prioritize multi-turn or specific conversation patterns.
– **Image analysis**: Use image metadata (resolution, content type) to guide selection.

Heavy-scaffolding: Self-research (rewriting)

- *Mandatory LLM rewriting after baselines*: Every post-baseline iteration must apply the MLLM provided as a tool to rewrite at least part of the training set (>20% samples recommended, >50% encouraged). Selection-only or programmatic-only iterations are explicitly banned.
- *Research-first mandate with eval-row grounding*: Similar to "self-research scaffolds", this version requires writing a research plan at each iteration, requiring the observation to cite specific failure rows from raw evaluation outputs (question, model output, gold answer), not just aggregate scores. A research plan that only references aggregated scores is explicitly declared insufficient.
- *Rewrite-recipe research questions*: a set of examples is provided as starting points: answer-length mismatches, format gaps, malformed samples, turn restructuring, ambiguous questions. These frame the kind of hypotheses the agent should form, distinct from previous selection-oriented thinking.
- *Rewrite verification step*: After curation/rewriting but before training, the agent must eyeball 5–10 rewritten samples to confirm the transform actually did what the hypothesis predicted. If the rewrite is generic or didn't change what was expected, the agent must fix the recipe before burning a training run. The result logs gains a 'rewrite recipe' column describing the specific LLM rewrite applied (target slice + transform).
- *Failure analysis section*: Requires reading raw evaluation files after scoring — sampling 5–15 wrong rows per benchmark, categorizing failures (OCR error, format violation, hallucinated option, etc.), and cross-referencing with training data to find the linkage between what the model learned and where it fails.
- *Revised no-contamination rule*: An explicit "No data contamination" rule forbids using evaluation questions or answers (from eval logs) to construct training examples — not as copies, paraphrases, or rewrite targets mirroring specific eval items. Patterns are fair game; specific items are not.

Adapt papers: EL2N top-loss with p95 assistant-loss noise filter (removes top 5% noisiest)

Data curation: ShareGPT4V-inspired response quality scoring.

Skill: sharegpt4v-improving-large-multi-modal-models-with-better-captions-arxiv-2311-12793v2

Adaptation: ShareGPT4V improves VLM training by ensuring high-quality captions/responses. We adapt this by decomposing the base model's forward-pass loss into total loss (EL2N informativeness) and assistant-only loss (response coherence quality). Select samples with high total loss (informative) but NOT extreme assistant-only loss (coherent responses). This filters out high-loss samples whose difficulty comes from noisy/incoherent responses rather than genuinely complex visual understanding.