HARNESS COMPILATION: WHICH DECISIONS SHOULD A SMALL VISION-LANGUAGE MODEL KEEP?

Anonymous authors
Paper under double-blind review

ABSTRACT

Small vision-language models (VLMs; $\leq$ 9B parameters here) are attractive for edge deployment but difficult to adapt with scarce supervision and fixed weights. External tools and memory can help, but using them introduces decisions that the model may not execute reliably. We introduce *Harness Compilation* (HC), an offline procedure that uses a large teacher to build a reusable harness for a frozen student. The teacher revises task knowledge and control logic from student execution traces, and a held-out validation set selects the harness. Deployment requires neither weight updates nor teacher calls. We organize runtime control into five operational degrees of freedom: invocation, selection, argument generation, evidence integration, and stopping or abstention. Interventions show that activation, open argument generation and evidence selection can be costly for the student, while several content rewrites and output constraints have little effect. Across seven visual question-answering settings, HC improves task scores by 12.0 to 22.4 points over bare models. Useful fact cards transfer to all ten evaluated students, but control policies are model-dependent and may need recompilation. With 100 practice items, HC achieves higher mean scores than answer-only LoRA on InfoSeek, DocVQA and SlideVQA. At larger budgets, weight adaptation can match or exceed the harness, while combining the two improves SlideVQA beyond either route alone. These results support adapting the division of work to the student rather than uniformly removing its decisions.

1 INTRODUCTION

Consider an industrial visual assistant that answers recurring image queries locally under limited memory and compute. For this edge setting, we focus on small vision-language models (VLMs), defined here as models with $\leq$ 9B parameters. Privacy or unreliable connectivity may rule out per-query cloud inference. Task-specific labels can also be scarce, and changing deployed weights may be inconvenient. We therefore study adaptation with fixed student weights and no large-model calls at deployment. A large teacher is available only offline to work with a limited set of practice examples.

An external *harness* provides instructions, task memories and execution procedures around the model. We focus on small executors whose ability to use these resources is uneven. A student may read a supplied fact correctly yet fail to request a lookup, select the right entry or generate a useful query. A harness that delegates these steps to the student can therefore impose control demands that exceed the executor's capabilities.

Our central hypothesis is that *the benefit of a harness depends not only on what it provides, but also on which decisions it leaves to the model*. Recent work automates harness construction through code search and execution feedback (Lee et al., 2026; Park et al., 2026; Yang et al., 2026), while Dynamo evolves skills and tools for frozen VLMs (Sun et al., 2026). We focus on the executor within such systems: *which decisions should a small VLM retain, and which should be compiled into external content or code?*

We introduce *Harness Compilation* (HC), an offline procedure that fits this division of work to a target student. A large teacher writes a candidate harness, runs it through the student's runtime on practice items, and revises it from execution reports. A separate validation set selects the retained version. The teacher acts as a compiler, while the frozen student executes the work that the harness

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

leaves to it (Figure 1). The reusable harness carries task knowledge and control outside the model weights.

We study five types of runtime decision: whether to invoke a resource, which candidate to select, what arguments to generate, how to integrate evidence, and when to stop or abstain. The objective is not to minimize the number of student decisions. On InfoSeek, retaining a bounded choice among candidate names helps, while asking the same student to generate an entity name is costly. On other tasks, a fixed rule or page selector is more useful. Changing the student can change the appropriate policy, even when the external content stays the same.

Across seven settings, HC improves scores over bare students with a few hundred practice items. Its benefits extend to the 100-item regime and can complement weight updates. Compilation also shifts work offline: in three content-task comparisons, the resulting runtime uses about 32 to 220 times fewer generated student tokens than chain-of-thought sampling with voting, while scoring higher. These are generation savings rather than end-to-end speedups.

Our contributions are:

- **Harness compilation with execution feedback.** We develop an offline procedure that writes and revises reusable content and control for a frozen student. Deployment requires no teacher calls or weight updates (Sections 4 and 5.2).
- **An empirical characterization of decision tolerance.** We organize runtime control into five decision types and measure how their value changes across tasks, interfaces and students. The results identify both useful student decisions and decisions better handled externally (Sections 5.3 and 5.4).
- **Limited-data adaptation compatible with weight updates.** We show strong performance with 100 practice items and evaluate when a harness complements, matches or is superseded by weight adaptation (Section 5.5).

2 Related work

Test-time scaling. Self-consistency samples and votes over reasoning paths (Wang et al., 2023). Adaptive visual inference allocates rollouts by query (Jeddi et al., 2026), and test-time augmentation combines predictions from transformed inputs (Kaya et al., 2026). Per-query visual programs and tool-using agents also place decisions in the inference loop (Surís et al., 2023; Gupta & Kembhavi, 2022; Qin et al., 2024). In contrast, HC fits reusable content and control offline. Our compute-only and student-driven tool controls test these alternatives using the target students and available resources (Appendices E.2 and E.6).

Prompt and context optimization. DSPy and MIPROv2 optimize instructions and demonstrations within a program (Khattab et al., 2023; Opsahl-Ong et al., 2024). GEPA uses reflective feedback (Agrawal et al., 2026), TextGrad provides textual critiques (Yuksekgonul et al., 2024), and ACE evolves a standing context (Zhang et al., 2026). We distinguish the search procedure from the object it can edit. HC exposes memory, routing and executable control as well as prompts. When GEPA edits the same harness, its performance is similar to our loop on the matched runs (Section 5.2). Our results therefore concern the decision structure being optimized, not a superior search algorithm.

Harness optimization. Meta-Harness and AutoSaddler revise harness code using execution evidence (Lee et al., 2026; Park et al., 2026). Related work adapts harnesses and contexts for smaller or video models (Qian et al., 2026; Yang et al., 2026; Xu & Chen, 2026). HARNESSEVO studies which harness components account for gains (Nguyen et al., 2026), while Lin et al. (2026) distinguish useful updates from a model's ability to activate and follow them. Other methods construct reusable tools or controllers (Cai et al., 2024; Zhang et al., 2024; Qiu et al., 2025). Dynamo evolves skills and executable tools for a frozen VLM (Sun et al., 2026). HC belongs to this line of work. Its focus is the allocation of runtime decisions between the student and external code, studied through interface interventions, cross-student transfer and recompilation after weight adaptation.

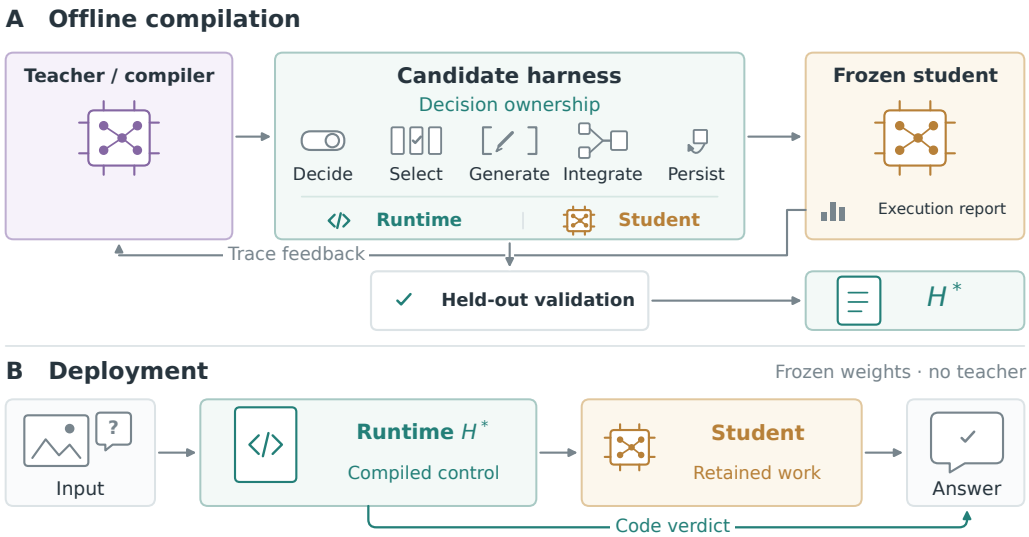


Figure 1: **Offline compilation, local execution.** The teacher revises a harness from student traces, and validation selects the deployed version. Code handles compiled decisions while the frozen student performs the retained work. The InfoSeek example retains a bounded candidate choice.

3 Problem: Which Decisions Should the Student Keep?

Setting. A task τ provides a labelled practice split D_p and a held-out test split D_t. A frozen student S answers under a harness $H = (P, T, R, M, C)$. Here P specifies prompts and an output contract, T supplies executable tools, R routes inputs and tool outputs, M stores retrievable content, and C constructs bounded candidate choices. An output contract specifies how to answer, including the answer format and whether abstention is allowed. The teacher L may revise these components offline, but the deployed harness cannot call L.

Five operational decisions. We use *degrees of freedom* (DoF) to describe the decisions exposed by a runtime. DoF-**decide** asks whether to invoke a tool or memory. DoF-**select** chooses among tools or entries. DoF-**generate** produces open arguments such as queries, names or code. DoF-**integrate** uses the supplied evidence to answer. DoF-**persist** determines stopping or abstention. These are categories of runtime control, not independent capability axes. A task need not expose all five, and changing one interface can affect another.

When should external evidence be used? Let a router assign a subset D_R to a channel c, such as a retrieved card or a tool verdict. Its *channel edge* is $\mathrm{acc}_c(D_R) - \mathrm{acc}_S(D_R)$, the channel's accuracy on those items relative to the bare student. A useful external channel needs a positive edge on a subset the router can identify. The relevant comparison is with the actual channel being deployed, not with the teacher that wrote the harness.

When should a decision stay with the student? We use the *retain criterion* $\pi_S(D_R) - \pi_0(D_R) - a_S(D_R) > 0$. Student decision precision π_S must exceed the mechanical alternative π_0 after accounting for the accuracy cost a_S of contract-induced abstentions. The abstention cost is not the raw abstention rate. This criterion motivates a measured comparison of interfaces rather than a rule to remove every student decision. Our experiments characterize these comparisons within the tested tasks and do not establish a universal decision ordering.

4 Harness Compilation

Construct, execute and revise. The teacher receives a task description, the editable harness interface and the bare student's practice score, then writes H_0. The runtime executes the student on

a probe subset of D_p and returns a report. The teacher proposes $H_{r+1} = L.\text{reflect}(H_r, \text{report}_r)$, adding, replacing or removing components. Main builds use three reflection rounds and no separate student profile (Appendix B.1).

Connect failures to editable decisions. Reports pair overall scores with diagnostics of individual stages. For InfoSeek, they separate accuracy with a correct entity card, a wrong card or no card; measure candidate-pick precision and abstention; and include 30 stratified failure traces with raw outputs. Wrong-card failures can motivate retrieval or routing changes. Failures despite a correct card can motivate changes to reading instructions or answer extraction. These practice-label diagnostics guide candidate edits, not fixed update rules. The teacher can request one additional trace query per round.

How feedback changes the division of work. In the InfoSeek 8B build, H_0 shows several cards without an explicit candidate pick. Near-zero accuracy on wrong-card items prompts the teacher to add a student pick on ambiguous retrieval (round 1). When traces show that this pick overturns correct top-1 links, the teacher narrows it to two candidates and favors top-1 (round 2). High-confidence links then bypass the pick (round 3): code decides when selection is needed, while the student resolves ambiguous cases. On Qwen3.5-2B, excessive abstention instead leads to removal of the "none" option. These revisions can expand or restrict the student's role. Appendix B.5 gives the round-by-round records.

Validation and deployment. A disjoint validation subset selects H^* from the candidate harnesses. Test items are never shown to the compiler. Small BLINK settings use three cross-build folds, each tested with a harness built on the other two. A later variant restarts a below-bare initial candidate and includes bare in validation selection (Appendix B.4). Deployment runs H^* with local tools and the frozen student, without teacher calls.

What the compiler can change. The editable artifact is a JSON specification of prompts, content and candidate-selection rules, with sandboxed Python hooks for routing and answer extraction. A rule may return a verdict without a student call. These are editable interfaces, not five sequential compiler passes (Appendix A).

5 Experiments

We evaluate task performance, decision allocation, transfer across students, weight adaptation and deployment cost.

Setup. The seven settings cover knowledge retrieval, document reading, visual perception, news images and multi-slide reading. Main results use Qwen3-VL-4B, except DocVQA, which uses Gemma-4-E4B. The transfer study includes ten students. The default teacher is GPT-5.5. Main builds use 150 probe and 300 validation items, except the smaller BLINK folds. Student weights are frozen and decoding is greedy unless a comparator states otherwise. DocVQA uses ANLS, SlideVQA uses exact match, and the other settings use accuracy. Tables 8 and 10 in Appendix A.1 record metrics, splits and build versions. Paired uncertainty and repeated builds are reported separately.

5.1 Overall effectiveness

HC improves over the bare student in all seven settings (Table 1). To test whether providing resources is enough, the *student-routed* control exposes the same resources with a plain instruction, without compiled routing, verdicts or output contracts. It recovers only part of the gain on several tasks and reduces depth accuracy below bare. The benefit of the compiled runtime is task-dependent. It is large on InfoSeek, SlideVQA, reflectance and depth, but not significant on LiveVQA or correspondence. These are whole-runtime comparisons rather than isolated effects of a single decision.

Strong fixed alternatives matter. Human recipes provide a more structured reference. HC exceeds these recipes on the content tasks, but no BLINK build significantly exceeds its fixed human rule. A further DocVQA control, an unoptimized typed-input/output program, reaches 86.2 and

Table 1: **Test performance with frozen students.** Higher is better. Average is the unweighted mean over the seven settings on their 0–100 task-score scales, not pooled accuracy. Human recipes are task-specific comparators, not matched-budget builds. Settings and recipe definitions are in Appendix A.1.

Setting	Bare	Student-routed	Human recipe	HC
InfoSeek	18.0	29.1	33.8	**40.2**
DocVQA	72.7	80.4	82.4	**85.6**
SlideVQA	44.7	44.4	48.7	**56.7**
LiveVQA	14.7	36.9	25.5	**37.1**
BLINK reflectance	50.8	50.0	59.0	**64.9**
BLINK correspondence	54.9	64.9	64.2	**67.6**
BLINK depth	82.3	73.4	94.4	**95.2**
Average	48.3	54.2	58.3	**63.9**

matches the compiler. Thus, a good fixed interface can already solve much of a task's adaptation problem. The role of compilation is to find useful content and control for the target student, not to guarantee an advantage over every manually designed harness (Appendices E.1 and E.5).

5.2 What does execution feedback contribute?

Table 2: **Initial and selected harnesses.** Each pair uses the same build; Average weights the seven settings equally. DocVQA uses the restart-rule build (83.0 to 85.2), while Table 1 retains the original 85.6 build. Further build details are in Appendix B.3.

Setting	Initial H_0	Selected H_{val}
InfoSeek	36.2	40.2
DocVQA	83.0	85.2
SlideVQA	41.9	56.7
LiveVQA	28.1	37.1
BLINK reflectance	59.7	64.9
BLINK correspondence	66.7	67.6
BLINK depth	95.2	95.2
Average	58.7	63.8

Table 2 shows how the selected harnesses change during compilation: the average test score rises from 58.7 to 63.8. Feedback improves InfoSeek, LiveVQA and SlideVQA, while depth keeps its initial rule. Independent one-shot builds also match the loop on some tasks, so repeated reflection is not uniformly necessary (Appendix B.2).

Traces help locate the failure. Aggregate scores do not reveal which decision or interface needs revision. Stage-specific diagnostics and raw outputs guide candidate changes (Appendix B.5). Compiler-input ablations compare traced and score-only feedback (Appendix B.1). These involve few builds and do not establish that any single feedback format is best for every task.

The editable artifact matters more than the search rule. Prompt-only search cannot revise the runtime's decision structure. When GEPA is instead allowed to edit the same executable harness with a fixed store, its mean score is similar to our loop on the two matched runs. Variation between builds exceeds that mean difference. This supports the value of exposing content and control to search, rather than a claim that our search algorithm is superior (Appendix E.4).

5.3 WHICH DECISIONS SHOULD THE STUDENT KEEP?

Table 3 first describes the selected programs. They do not all minimize student involvement. InfoSeek retains a bounded candidate choice, LiveVQA leaves all three cards for the student to read, and BLINK often lets code answer directly. Table 4 then replaces one part of each interface while leaving the rest of the selected harness in place.

Table 3: **Division of work in the selected harnesses.** These are task-specific roles, not capability scores. The full decision map is in Appendix C.2.

Setting	Compiled work	Student retains
InfoSeek	Route retrieval and prepare a card	Choose among names on ambiguous inputs, then answer
DocVQA	Supply OCR and an answer format	Read the page and supplied text
SlideVQA	Rank and select evidence slides	Read selected slides and answer
LiveVQA	Supply three article cards	Select and combine evidence while answering
BLINK	Compute a verdict from measurements	Answer when the rule defers

Table 4: **Effect of replacing one decision interface.** Entries are score changes relative to the selected HC build. $^{\text{ns}}$ denotes $p > 0.05$, and n/a denotes an untested intervention. Definitions and earlier-build results are in Appendix C.1.

Setting	Decide	Select	Generate	Integrate	Persist
InfoSeek	-9.7	-2.2	-13.8	-0.3^{ns}	-5.5
DocVQA	-4.2	$+0.5^{\text{ns}}$	n/a	-1.1	-0.5^{ns}
SlideVQA	$+0.5^{\text{ns}}$	-15.4	n/a	-2.0	-0.4^{ns}
LiveVQA	n/a	n/a	n/a	$+2.2$	-3.0
BLINK reflectance	$+0.0^{\text{ns}}$	n/a	n/a	-14.9	-0.8^{ns}
BLINK correspondence	-2.5	n/a	n/a	-2.7^{ns}	-3.4
BLINK depth	$+0.0^{\text{ns}}$	n/a	n/a	-21.8	-0.8^{ns}

Decide: compile activation when requests are unreliable. Requiring the student to request a useful resource costs performance on InfoSeek and DocVQA. The corresponding changes are not significant on SlideVQA, reflectance or depth. Activation should therefore be compiled where the request interface is unreliable, not simply because the student is small.

Select and generate: bounded choices differ from open arguments. On InfoSeek, the student improves ambiguous retrieval by choosing among a few candidate names (Appendix D.1). Asking it to generate the entity name instead produces a much larger loss. Importantly, the select intervention does not merely swap code for the student. It replaces the explicit bounded-choice step with all candidate content in one context. On SlideVQA, removing the slide-selection mechanism is especially costly. These results support carefully designed selection interfaces, not a blanket instruction to compile selection out.

Integrate: the evidence channel determines the difficulty. Replacing rewritten cards or OCR with raw content has small effects on the content tasks and improves LiveVQA. On BLINK, the intervention is different: it replaces the processed rule output with raw measurements. The large losses on reflectance and depth show the value of converting those measurements into a usable verdict. They do not imply that every form of evidence integration should be removed from the student.

Persist: contracts matter, but not in the same way everywhere. Removing the answer contract and parser hurts InfoSeek and LiveVQA, but has little effect on DocVQA and SlideVQA. On BLINK, showing a verdict as a suggestion is almost as effective as returning it directly on reflectance

and depth. The useful computation is largely the rule that produces the verdict. Abstention also depends on the metric. The LiveVQA harness and student-routed control have similar accuracy, but the latter scores better under an abstention-sensitive F-score (Appendix D.4).

Each intervention is a specific, unselected interface variant compared with a validation-selected harness. Their effects are not additive, and nonsignificance does not establish equivalence across all possible prompts. Together they support measuring the student's decisions in context rather than imposing a universal policy.

5.4 Student-dependent decisions and recompilation

Content can transfer when a decision does not. Teacher-written InfoSeek cards benefit all ten evaluated students. The bounded selection policy is less portable. MiniCPM picks no better than the mechanical linker, while Qwen3.5-2B frequently takes the "none" option and discards potentially useful evidence. Selection precision without abstentions therefore gives an incomplete picture. The diagnostics in Appendix C.3 characterize these differences, but are not an independent prospective prediction test.

Recompile the policy for the receiving student. Figure 2 compares unchanged transfer with a fresh build. Each receiving student has an own build above the InfoSeek transplant, although the 8B's first build fails and its second succeeds. In DocVQA, an OCR harness compiled for Gemma harms stronger document readers. Fresh builds detect that the student is copying noisy OCR, remove the tool and return near bare performance. This illustrates the channel-edge criterion: a stronger teacher does not make a weaker OCR engine a useful channel for every student.

A fixed verdict can also become inappropriate. BLINK code returns the same verdict regardless of the student, but its benefit depends on the student's own accuracy. On deferred items, the receiving model still has to answer under the supplied interface. A fresh correspondence build for the 2B removes that fallback and improves over the transplant. Updating the student's weights can change the same tradeoff, which we examine next.

5.5 Data efficiency and weight adaptation

Table 5: **Adaptation with 100 practice items.** Mean test scores over three runs. Training columns use the stronger reported epoch variant where available. Full schedules and individual builds are in Appendix F.1.

Setting	HC	Answer LoRA	Trajectory SFT
InfoSeek	**36.2**	23.3	35.3
DocVQA	**84.4**	81.1	n/a
SlideVQA	**53.7**	47.3	n/a

Useful adaptation with little task data. Table 5 uses 50 probe and 50 validation items for HC. Its mean score exceeds answer-only LoRA on all three tasks. The advantage is not uniform across builds, and longer trajectory training nearly closes the InfoSeek gap. At 450 items, trajectory SFT exceeds the original InfoSeek HC builds but falls below the later 40.2 build. At 1,200 items, the two routes are not significantly different. These results show a useful low-data alternative, not universal superiority over training.

The routes can be complementary or redundant. On SlideVQA, combining a harness with a tuned student outperforms either alone (Table 6). On DocVQA, adding the harness to a 999-label adapter gives no further gain. With enough external reflectance labels, LoRA surpasses the frozen harness. The old harness then reduces performance, while recompilation removes most of the harmful control and recovers the tuned student's score. Detailed scaling and matched-label comparisons are in Appendix F.

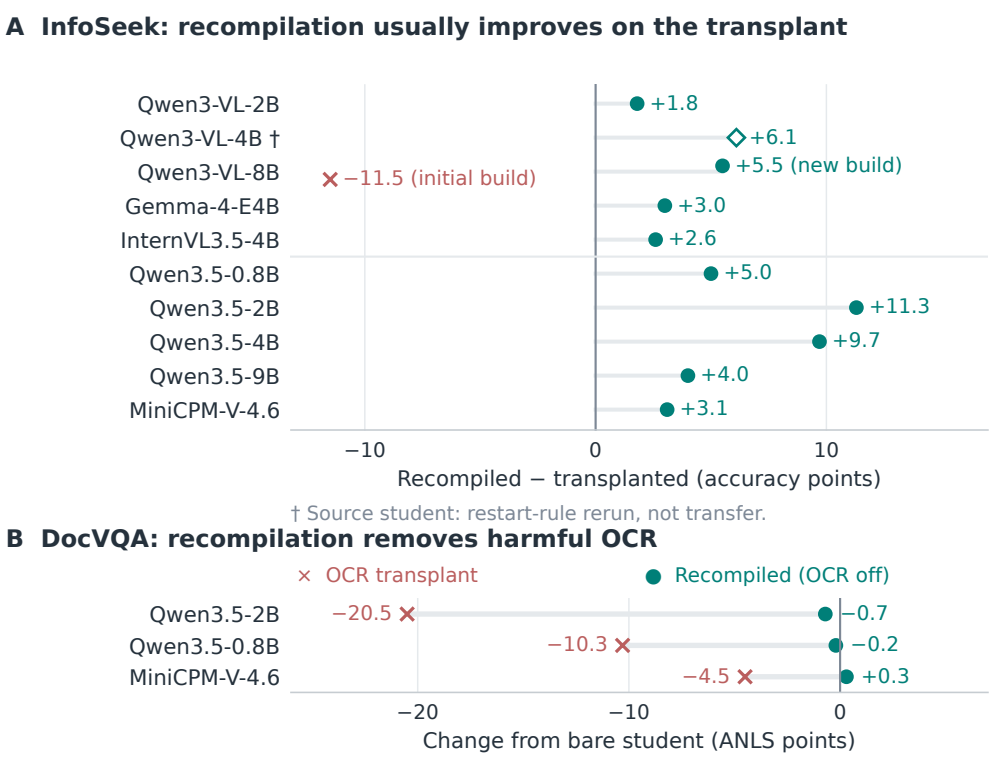


Figure 2: **Recompilation can repair a transferred policy.** InfoSeek shows improvements and an 8B failure. DocVQA builds remove harmful OCR. The hollow 4B point is a same-student rerun, not transfer. Full protocols are in Appendix C.3.

Table 6: **Combining HC with weight adaptation.** The same harness is evaluated with the base and tuned student. Component budgets and training protocols are in Appendix F.3.

Setting	LoRA labels	HC alone	LoRA alone	LoRA + HC
SlideVQA	1,300	56.7	53.2	**63.9**
DocVQA	999	84.9	84.7	84.7

Matched practice items do not mean identical supervision. HC uses a teacher and external resources, while answer-only LoRA receives gold answers directly. The gate consumes labels for selection rather than gradient updates. The combination experiments establish compatibility, but do not identify exactly which skills entered the weights.

5.6 Generation efficiency and offline cost

HC shifts work from online generation into reusable content and code. Compared with chain-of-thought sampling and voting, it uses about 32 to 220 times fewer generated tokens on the three content tasks while scoring higher (Table 7). On BLINK, direct code verdicts further reduce student calls. CoT with voting matches HC on reflectance and is not significantly different on correspondence, but requires much more generation.

Token counts include every sampled rollout and student pass. They exclude image and input prefill, tool computation and offline teacher use, so the ratios are not end-to-end speedups. Appendix E.2 specifies the sampling settings and Appendix G.3 separates the cost categories.

Table 7: **Performance and generated student tokens at deployment.** Each row uses the same student and test items. InfoSeek and DocVQA use the earlier builds with recorded token counts (Appendix A.1).

Setting	Test score ↑		Tokens per item ↓	
	CoT + vote	HC	CoT + vote	HC
InfoSeek	24.6	39.7	1,120	5
DocVQA	80.4	84.9	580	11
SlideVQA	51.1	56.7	970	30
BLINK reflectance	64.9	64.9	2,000	0.1
BLINK correspondence	68.5	67.6	2,510	0.7
BLINK depth	85.5	95.2	2,080	0.1

Offline cost. Reported VQA builds cost about $0.3 to $7 in teacher calls, with 1 to 35 student GPU-minutes. For InfoSeek, about $1 covers compilation with an existing store. Creating the reusable fact store adds about $6. These are different cost regimes, and neither requires teacher calls during deployment.

6 Discussion and limitations

Decision allocation, not uniform restriction. The results favor retaining decisions that the student executes well and moving unreliable work into an external interface. Useful content, a bounded choice and a code verdict play different roles. Our taxonomy organizes these roles, but the interventions do not establish independent capability axes or a general prediction guarantee. A good fixed interface can match compilation, and another search method can find similar harnesses.

Build reliability. A small validation set can select a candidate that is worse on test. Initial teacher draws also vary. The main InfoSeek build uses the restart-rule procedure but succeeds on its first draw, so its improvement over the original build cannot be attributed to restarting. Failed builds and earlier results remain in Appendix B. BLINK has small test pools and limited build replication. The support audit also finds rules that fit probe phrases without affecting test performance (Appendix G.1).

Scope of the evidence. Residual teacher gaps can reflect missing tools, retrieval failures or interface limitations. They do not prove that reasoning cannot be compiled. Our experiments measure task performance and generation costs, not end-to-end industrial deployment or hardware-specific latency. Offline teacher access to practice traces is still required. The exploratory LIBERO-plus study primarily shows instruction repair rather than long-horizon planning and is kept outside the seven-setting VQA evaluation (Appendix G.2).

7 Conclusion

Small VLMs need harnesses designed for the decisions they can execute reliably. Harness Compilation makes this allocation an offline adaptation problem: a teacher uses student traces to build reusable knowledge and executable control, while the student retains useful bounded decisions. Across seven VQA settings, HC raises the macro-average task score from 48.3 to 63.9, with gains of 12.0 to 22.4 points over bare students and no weight updates or teacher calls at deployment. Interface interventions identify costly activation, argument-generation and evidence-use decisions, while also showing the value of retaining a reliable student choice. Transfer and weight adaptation change which allocation works, making recompilation part of adapting the system to its executor. HC provides an effective route with limited task data and can complement training when the two supply different benefits. The broader lesson is to adapt the interface around a model as deliberately as the model itself, using measured execution behavior to decide which work to retain and which to compile.

AI USE STATEMENT

We used GPT-6 to search for and identify related work. Fable 5.1 assisted research execution, including running experiments, and produced initial drafts of paper sections under author supervision. GPT-6 Astra helped revise the manuscript's organization and wording to improve readability. The workflow also included AI feedback on experimental design and interpretation of results, editing of figure-generation code, and bibliographic checks; the authors made the final methodological and editorial decisions.

Generative models also form part of the reported experimental procedures. GPT-5.5 and the additional teachers evaluated in Appendix B.1 generate harness content and code offline. GPT-5.5 produces reusable fact cards and synthetic tool-use trajectories for the trajectory-SFT baselines (Appendices A and F.2). Claude Sonnet 5 judges LiveVQA answers, with GPT-5.5 used in the separate scoring audit (Appendix D.4). These uses are accounted for in the corresponding protocols and cost reports. Benchmark questions and reference answers are taken from the evaluation datasets. The deployed HC runtime makes no teacher calls.

The authors directed the research and supervised the experimental workflow. They take responsibility for the experimental design, interpretation, citations and final manuscript, including all AI-assisted text, code and generated artifacts. This empirical study does not include AI-generated mathematical proofs.

REPRODUCIBILITY STATEMENT

Task splits, scoring, runtime interfaces and verbatim InfoSeek prompts are provided in App. A. Compiler ablations and failed builds are in App. B; decision interventions and transfer diagnostics are in App. C. Task-specific results, baseline configurations and training budgets appear in Apps. D–F. App. G records support audits, evaluation safeguards and cost accounting. We distinguish reused stores, different compiler procedures and training-label budgets throughout these comparisons.

REFERENCES

Lakshya A. Agrawal et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.19457.

Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics (SIGGRAPH)*, 2014.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.17126.

Yang Chen et al. Can pre-trained vision and language models answer visual information-seeking questions? *arXiv preprint arXiv:2302.11713*, 2023.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025. URL https://arxiv.org/abs/2510.13626.

Mingyang Fu, Yuyang Peng, Dongping Chen, Zetong Zhou, Benlin Liu, Yao Wan, Zhou Zhao, Philip S. Yu, and Ranjay Krishna. Seeking and updating with live visual knowledge. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2504.05288. Datasets and Benchmarks Track; arXiv:2504.05288.

Xingyu Fu et al. BLINK: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision (ECCV)*, 2024. arXiv:2404.12390.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. *arXiv preprint arXiv:2211.11559*, 2022.

Edward J. Hu et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Ahmadreza Jeddi, Minh Ngoc Le, Amirhossein Kazerouni, Hakki Can Karaimer, Hue Nguyen, Iqbal Mohomed, Michael Brudno, Alex Levinshtein, Konstantinos G. Derpanis, Babak Taati, and Radek Grzeszczuk. AVIS: Adaptive test-time scaling for vision-language models. *arXiv preprint arXiv:2606.11576*, 2026.

Mehmet Onurcan Kaya, Desmond Elliott, and Dim P. Papadopoulos. Efficient test-time scaling for small vision-language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.03574.

Omar Khattab et al. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Zihang Lai, Senthil Purushwalkam, and Abhinav Gupta. The functional correspondence problem. In *ICCV*, 2021.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving LLM agents. *arXiv preprint arXiv:2605.30621*, 2026.

Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local feature matching at light speed. In *ICCV*, 2023.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *WACV*, 2021.

Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. SPair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019.

Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. *arXiv preprint arXiv:2306.09683*, 2023.

Michael Nguyen, Wei Chen Tan, Nurul Aisyah Hassan, Arvind Raman, Li Hua Lim, and Ahmad Faiz Razak. Where does harness-optimization value live? Localized gains and the budget-splitting trap in self-evolving LLM agents. *arXiv preprint arXiv:2609.02889*, 2026.

Krista Opsahl-Ong et al. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*, 2024.

Maxime Oquab et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Sungho Park, Wonjoong Kim, Rongyuan Tan, Jue Zhang, Wook-Shin Han, Pengfei Gao, Chanyoung Park, Yongqiang Yao, Rao Fu, Elsie Nallipogu, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. AutoSaddler: Automatic harness optimization with durable updates from agent execution traces. *arXiv preprint arXiv:2608.23041*, 2026.

Cheng Qian, Wenting Zhao, Liangwei Yang, Heng Wang, Jielin Qiu, Heng Ji, Silvio Savarese, Huan Wang, and Shelby Heinecke. AI4AI at test-time: Strong-to-weak capability transfer via harnesses. *arXiv preprint arXiv:2608.12307*, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, Shilong Liu, Xun Jiang, Liu Leqi, and Mengdi Wang. AgentDistill: Training-free agent distillation with generalizable MCP boxes. *arXiv preprint arXiv:2506.14728*, 2025.

Yutao Sun, Yanting Miao, Hao-Xuan Ma, Mengyu Zhou, Mingshuai Chen, Tiancheng Zhao, Dexin Wang, Lei Lv, Li Xu, Xiaoxi Jiang, and Guanjun Jiang. Dynamo: Dynamic skill-tool evolution for vision-language agents. *arXiv preprint arXiv:2606.30185*, 2026.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. SlideVQA: A dataset for document visual question answering on multiple images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 13636–13645, 2023. doi: 10.1609/aaai.v37i11.26598. URL https://ojs.aaai.org/index.php/AAAI/article/view/26598.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Guoyang Xu and Hao Chen. VideoHarness-RSI: Recursive harness self-improvement for long-video understanding with frozen vision-language models. *arXiv preprint arXiv:2608.24302*, 2026.

Yibin Yan and Weidi Xie. EchoSight: Advancing visual-language models with wiki knowledge. *arXiv preprint arXiv:2407.12735*, 2024.

Chenyang Yang, Xinran Zhao, Tongshuang Wu, and Christian Kästner. Better harnesses, smaller models: Building 90% cheaper agents via automated harness adaptation. *arXiv preprint arXiv:2607.08938*, 2026.

Lihe Yang et al. Depth Anything V2. *arXiv preprint arXiv:2406.09414*, 2024.

Mert Yuksekgonul et al. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Qizheng Zhang et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.04618.

Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Offline training of language model agents with functions as learnable weights. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.11359.

Appendix Contents

Organization. The seven sections below group implementation, controlled comparisons, task-specific evidence, and reliability checks.

A Implementation, prompts and evaluation protocols . . . 14
A.1 Tasks, splits and evaluation . . . 14
A.2 Harness languages . . . 16
A.3 Student profile protocol . . . 17
A.4 Verbatim prompts and an executable harness . . . 17
B Compiler ablations and build stability . . . 28
B.1 Teacher, profile and feedback ablations . . . 28
B.2 One-shot construction . . . 28
B.3 Build variation and validation selection . . . 28
B.4 Failed initial builds and the restart safeguard . . . 29
B.5 Execution feedback and decision allocation . . . 31
C Decision diagnostics and cross-student transfer . . . 31
C.1 Student-routed and single-decision controls . . . 31
C.2 Decision ownership and student-specific policies . . . 33
C.3 Cross-student transfer and recompilation . . . 33
C.4 Evidence for decision constraints and remaining reasoning gaps . . . 36
D Additional VQA results . . . 38
D.1 InfoSeek: content, placement and store lineage . . . 38
D.2 DocVQA: consolidated baseline comparison . . . 40
D.3 BLINK: verdict coverage and task-dependent residuals . . . 40
D.4 LiveVQA: retrieval, contracts and scoring . . . 40
D.5 SlideVQA: page selection and image budget . . . 42
E Comparisons with alternative adaptation methods . . . 43
E.1 Fixed and tuned human harnesses . . . 43
E.2 Test-time scaling . . . 44
E.3 Retrieval and prompt optimization . . . 45
E.4 GEPA over the same executable artifact . . . 46
E.5 Typed input and output fields . . . 46
E.6 Student-driven code and tool-use controls . . . 46
F Data efficiency and weight adaptation . . . 48
F.1 Practice budgets and the 100-item regime . . . 48
F.2 Trajectory supervision at matched item budgets . . . 48
F.3 Answer adaptation and task-specific training protocols . . . 48
F.4 Label scaling and recompilation after tuning . . . 51
G Reliability audits and cost . . . 52
G.1 Support, store and evaluation audits . . . 52
G.2 Exploratory VLA boundary case: LIBERO-plus . . . 52
G.3 Compilation and deployment cost . . . 53

A IMPLEMENTATION, PROMPTS AND EVALUATION PROTOCOLS

A.1 TASKS, SPLITS AND EVALUATION

How to read the main comparisons. Table 8 records the evaluation settings used in the main results. Table 9 identifies the human comparator in each row. Table 10 distinguishes the main builds from the reference builds used in earlier comparisons. Teacher scores are listed separately in Table 11 because their evaluation populations are not uniform.

Table 8: Main evaluation settings. Practice counts are probe / validation items.

Setting	Student	Metric	Test items	Practice
InfoSeek	Qwen3-VL-4B	Relaxed exact match	6,000	150 / 300
DocVQA	Gemma-4-E4B	ANLS	3,568	150 / 300
SlideVQA	Qwen3-VL-4B	Exact match	2,215	150 / 300
LiveVQA	Qwen3-VL-4B	Judged accuracy	2,000	150 / 300
BLINK reflectance	Qwen3-VL-4B	Accuracy	134	44 / 45 per fold
BLINK correspondence	Qwen3-VL-4B	Accuracy	441	147 / 147 per fold
BLINK depth	Qwen3-VL-4B	Accuracy	124	41 / 42 per fold

Main builds use GPT-5.5, no separate student profile and three reflection rounds. BLINK uses three cross-build folds. Each test fold is evaluated with a harness built on the other folds. The listed BLINK counts retain the reported per-fold allocations. InfoSeek's main build uses the restart-rule procedure, but its first candidate passed the probe floor and no restart occurred. DocVQA's main result is the original 85.6 build, not the 85.2 restart-rule rerun. LiveVQA uses April and May 2025 news images and the Claude Sonnet 5 judge. The judge did not participate in compilation. Appendix D.4 separates judged accuracy, containment and the abstention-sensitive F-score.

Table 9: Human recipes used in Table 1. Full prompts are in Appendix E.1.

Setting	Human recipe	Score
InfoSeek	Select a candidate name, then read retrieved Wikipedia passages	33.8
DocVQA	Supply page OCR in typed fields	82.4
SlideVQA	Read the three slides selected by the text ranker	48.7
LiveVQA	Select one candidate headline and read its article card	25.5
BLINK reflectance	Fixed Retinex verdict rule	59.0
BLINK correspondence	Fixed LightGlue and DINO rule	64.2
BLINK depth	Fixed depth-estimator rule	94.4

These recipes are fixed or development-tuned references rather than a matched-budget comparison with HC. The InfoSeek recipe uses public Wikipedia content, not the teacher-written fact store. The tuned InfoSeek placement policy over the shared teacher store scores 37.4 and is a separate reference. Likewise, the unoptimized DSPy DocVQA program scores 86.2 and is distinct from the typed-field recipe at 82.4. The student-routed controls use the resources supplied to HC with a plain instruction. Their exact interfaces are defined in Appendix C.1.

The two InfoSeek transfer analyses use different source harnesses. The recompilation comparison fixes a 450-item source, while the selection diagnostics use the earlier 1,200-item source. Appendix C.3 separates them. On SlideVQA, Table 6 combines the 450-item harness at 56.7 with a 1,300-label LoRA. On DocVQA it combines the 999-item harness at 84.9 with the 999-label LoRA. These are component comparisons, not a common total adaptation budget across tasks.

Table 10: Build identities across comparisons. Counts are unique practice items.

Comparison	InfoSeek reference	DocVQA reference
Main results and main interventions	40.2, 450 items, restart-rule procedure	85.6, 450 items, original procedure
Initial-to-selected comparison (Table 2)	36.2 to 40.2, same main build	83.0 to 85.2, 450 items, restart-rule rerun
Generated-token comparison (Table 7)	39.7, 1,200 items	84.9, 999 items
One-shot construction comparison	39.7, 1,200 items	84.9, 999 items
InfoSeek recompilation comparison	Original 34.1 source at 450 items	Not applicable
InfoSeek decision diagnostics	Earlier 1,200-item source with its own rewritten store	Not applicable
Representative runtime-cost records (Table 43)	39.7 build	85.7 build

Table 11: Teacher reference scores. Different populations prevent treating this column as a common upper bound.

Setting	GPT-5.5	Evaluation population and input
InfoSeek	50.0	300-item practice subset
DocVQA	93.3	Full test set, 2,048-pixel images
SlideVQA	64.7	150-question practice pool with the linker's top-three slides
LiveVQA	58.3	300-question practice pool with the same candidate passages
BLINK reflectance	79.9	Same items as the student and harness
BLINK correspondence	78.7	Same items as the student and harness
BLINK depth	89.5	Same items as the student and harness

The seven primary evaluation settings cover knowledge retrieval (InfoSeek (Chen et al., 2023; Yan & Xie, 2024)), document reading (DocVQA (Mathew et al., 2021)), three perceptual settings from BLINK (Fu et al., 2024), news-image question answering (LiveVQA (Fu et al., 2025)), and multi-page reading (SlideVQA (Tanaka et al., 2023)). InfoSeek uses 3,000 practice and 6,000 test items; DocVQA reports ANLS on 3,568 test items. LiveVQA and SlideVQA splits and scorers are specified in Apps. D.4 and D.5. Unless stated otherwise, a build uses 150 probe items, 300 disjoint validation items and three reflections. The probe supplies labelled execution traces; validation scores select the retained harness and are visible in the round history. Final test items are not supplied to the compiler. BLINK uses three cross-build folds, with each item evaluated only by a harness built on the other folds. Practice counts include both probe and validation items; accepted trajectory counts are reported separately from item budgets (App. F.1).

Why these BLINK tasks. An exploratory survey covered all 14 BLINK tasks. We retain five tasks grouped into three settings (Table 12) because they expose distinct relationships between available measurements and student decisions: weak photometric evidence, geometric matching with semantic residuals, and a strong depth channel. This is a study of these interfaces, not a claim of coverage across all BLINK capabilities. A large teacher–student gap alone was not an inclusion rule: depth has a modest teacher gap but a useful external estimator, whereas IQ Test showed no suitable carrier among the resources screened. The latter remains a diagnostic in App. C.4.

Statistical reporting. Binary correctness contrasts use paired exact McNemar tests on the same items. ANLS score contrasts use paired item bootstrap intervals; analyses that threshold ANLS at 0.5 are identified explicitly. Intervals and nonsignificant contrasts are reported where available. Paired differences retain their original precision before the displayed scores are rounded. Repeated builds are shown individually: variation across teacher draws is distinct from uncertainty over test items. Different prompt contracts, stores, image resolutions and validation budgets are identified rather than pooled as identical runs.

Tools. BLINK reflectance uses IIW-derived images (Bell et al., 2014); correspondence includes SPair-71k (Min et al., 2019) and FunKPoint (Lai et al., 2021). Available signals come from DINOv2 (Oquab et al., 2023), LightGlue (Lindenberger et al., 2023), Depth-Anything-V2 (Yang et al., 2024), and the task-specific measurements below. OWLv2 (Minderer et al., 2023) is used in the exploratory policy evaluation (App. G.2); weight comparisons use LoRA (Hu et al., 2022).

A.2 HARNESS LANGUAGES

Knowledge language (InfoSeek). Keys: `prompt` (system, before/after question, card frame with {`cards`}, answer suffix), `memory` (k, `sim_floor`, `max_card_chars`, `store` v1/v2), `decode.max_new_tokens`, `discriminate_prompt` (with {`options`} and a "0. None of them" option), `name_prompt`, `router_py` (`def select(item)` $\rightarrow$ mode `answer`/`discriminate`/`name`, cards to show, note), `parse_py`, and a store rebuild request honoured once per build (1,380 cards, written from entity names only; the teacher never sees questions or answers when writing cards). Discriminate runs as two passes in separate contexts: the student picks a number, then answers with the chosen card; "none" answers without a card.

Table 12: BLINK task selection: the five selected tasks form three evaluation settings.

Setting	Items	What the comparison tests
Relative reflectance	134	Whether local photometric measurements can support a useful verdict despite incomplete illumination invariance.
Correspondence	441	Visual (172), semantic (139), and functional (130) correspondence share a matching interface but differ in what geometric signals can resolve.
Relative depth	124	Whether a depth estimator's measurements are better used through code than interpreted by the student.

Document language (DocVQA). `prompt` (system, before/after, `tool_frame` with {`tool_text`}, suffix), `tool` (`ocr` on/off, `max_chars`), `decode`, `router_py` (`def select(item)` → tool text or none, note), `parse_py`.

Perception language (BLINK). `prompt`, `tool.signals` on/off, `decode`, `answer_mode` `student|verdict_if_present`, `router_py` (`def select(item)` → tool text, verdict letter, note), `parse_py`. Signals per bed: reflectance — marker coordinates, RGB at radius 3/8, chromaticity, HSV, luminance, context luminance in 15/40-px annuli, Retinex log-reflectance at σ 5/15/40; correspondence — DINOv2 patch cosine (3×3 and 1×1) between REF and each candidate, best-match location and its distance to candidates, SuperPoint+LightGlue projection of REF and its distances, match count; depth — Depth-Anything-V2 Small and Base at radius 4/10, local std, percentile, vertical position, per-model A−B difference.

A.3 Student profile protocol

Table 13: Student-profile measurements on practice subsets. Greedy decoding, 300 items unless stated otherwise.

Measure	Qwen3-VL 2B	Qwen3-VL 4B	Qwen3-VL 8B	Gemma-4 E4B	InternVL3.5 4B
A1 activation rate / name precision	0.0 / 0.0	82.3 / 30.4	89.3 / 36.2	97.3 / 15.1	58.7 / 0.6
A2 compliance t1 / t6	36 / 32	100 / 100	98 / 98	100 / 90	22 / 18
A3 text / image demos Δ	−7.0 / −0.7	−7.3 / −2.7	+1.0 / +2.3	+1.7 / 0.0	−2.7 / +2.0
A4 correct, menu 1 / 3 / 7	19 / 10 / 6	66 / 38 / 30	75 / 52 / 37	46 / 37 / 27	45 / 48 / 39
A4 over-fire at 7	87	98	91	36	46
A5 valid JSON 1 / 6 fields	100 / 100	100 / 99.7	100 / 99	99.7 / 99	99 / 27.7
A6 identical resend / answered directly	22.5 / 61.9	0.5 / 99.5	0.5 / 93.8	0.0 / 100	3.4 / 71.5
A7 frame cost	−1.3	−8.0	−3.3	+0.3	+0.7
A8 k=1 / k=3 Δ	+1.7 / −10.7	+5.3 / −2.7	+5.0 / +7.7	+11.7 / +2.7	+5.3 / +3.7
card capture fraction	38	37	37	59	59
A9 discrimination precision when picking	57.0	66.0	61.2	49.3	50.8
tool-offer ask rate (DocVQA / ChartQA)	–	1.0 / 4.0	–	0.0 / 0.4	–

The profile separates activation and name precision (A1), one-line rule compliance at turns 1 and 6 (A2), text and image demonstrations (A3), menus of 1, 3 and 7 tools (A4), JSON with 1 and 6 fields (A5), resending after failure and answering directly (A6), response framing (A7), one-card versus three-card input (A8), and candidate discrimination (A9). Card capture is reported as a fraction. The OCR offer rates use separate DocVQA and ChartQA contracts.

Activation rates depend on the contract: Gemma activates memory under one lookup interface but does not request OCR under an offer interface. The profile separates syntax compliance, argument precision, selection and response framing; no single score is treated as a general reasoning-capability scale. The compiler-input ablation (App. B.1) tests whether these measurements help when supplied as sensitivities or as a menu of executable decisions.

A.4 Verbatim prompts and an executable harness

Compile and reflect prompts. The compile prompt contains the language documentation, the task description, the bare probe score and, when one is measured, the student profile (the main-table builds are no-profile builds, so that block reads "no profile measured"). The reflect prompt restates the documentation, the history of rounds with validation scores and gate decisions, the current harness JSON and the traced report of the last round; the teacher may reply with one trace query before writing the next harness. The seven listings below give, for InfoSeek, the full compile

template (Prompt 1), the reflect template with the head of a real round report (Prompt 2), the fact-card generation prompt with the instructions the teacher wrote for the store used by the main-table build (Prompt 3), the two student calls of one test item under the main-table harness (Prompt 4), and that harness itself: its JSON (Prompt 5), router (Prompt 6) and parser (Prompt 7). Placeholders in angle brackets are filled by the runtime; everything else is verbatim. The DocVQA, SlideVQA, BLINK, LiveVQA and LIBERO-plus languages are released with the code.

Prompt 1: Teacher compile prompt for InfoSeek (round 0), as sent for a no-profile build. The language documentation is the first block; the task description, the profile block, three store examples and the bare probe score follow.

```
You are compiling an INFERENCE-TIME HARNESS for a small frozen vision-language model (the
     student). The harness is a JSON object; the runtime executes it. The student is called
     with an image and a text prompt and must answer a knowledge question about the entity in
      the image (InfoSeek: e.g. "What is the elevation of this mountain?", "Which family does
      this plant belong to?"). Answers are short strings or numbers; scoring is the official
     InfoSeek relaxed exact match (numbers must fall in the gold range).

At inference NO large model is available. The only deterministic signals the runtime can
     compute per item are:
- clip: the top-3 entity names from a 1,380-entity store (the gold entity is always in the
     store), each with a CLIP ViT-B-32 image->name cosine similarity (typically 0.2-0.4; top
     -1 is the gold entity about 52% of the time, top-3 contains it 68%), and that entity's
     memory card text.
- qtype: "num" if the question expects a number, else "str"; split: "unseen_question" or "
     unseen_entity" (InfoSeek's two test conditions; no other meaning here).
- The question text.
Cards: the store's current cards are mostly just the entity NAME (median 17 characters); only
      138 of 1,380 are full fact cards. You may request a regenerated store ("memory.store":
     "v2", written by you offline with your own instructions; its cost is charged to this
     build)  -  see "rebuild_store" below.

HARNESS JSON (all keys optional; defaults shown):
{
 "name": "short id", "rationale": "why this harness, 1-3 sentences",
 "prompt": {
   "format": "plain",                  # or "typed_fields": the DSPy-adapter serialisation (
     system message declaring typed input/output fields, [[ ## field ## ]] markers, answer
     field + completed marker; answer_suffix becomes the objective sentence; the output is
     parsed from the answer field unless parse_py is set). It is a fixed contract, not an
     optimiser; on DocVQA it was worth +8 bare / +3 with OCR for Gemma.
   "system": null,                      # a system message; the student profile says standing
      advice blocks HURT this student
   "before_question": "",               # text placed before the question
   "after_question": "",                # text placed after the question / cards
   "card_frame": "Retrieved from your memory store (may or may not describe the right entity)
     :\n{cards}",   # must contain {cards}
   "answer_suffix": "Answer the question with a single word, number, or short phrase."
 },
 "memory": {"k": 1, "sim_floor": 0.0, "max_card_chars": 600, "store": "v1"},   # k in 0..3 =
     how many CLIP top cards are shown by default (subject to the floor)
 "decode": {"max_new_tokens": 64},
 "discriminate_prompt": "Which of these is the entity shown in the image?\n{options}\n0. None
      of them\nAnswer with the number only.",   # used when a router returns mode "
     discriminate" (student picks 1..n or 0 = no card); must contain {options}
 "name_prompt": "What is the specific name of the entity ... Answer with the name only.",   #
      used when a router returns mode "name" (student names the entity; MiniLM links the name
      into the store; the profile says this student names correctly ~30% of the time)
 "name_sim_floor": 0.5,
 "router_py": null,    # OPTIONAL python source defining  def select(item) -> dict  ; runs in
      a sandbox (stdlib only, no network, no files)
 "parse_py": null      # OPTIONAL python source defining  def parse(text) -> str   ; extracts
      the final answer from the student's raw output
}

router select(item) receives item = {"uid", "question", "qtype", "split", "clip": [{"name","
     sim","card","entity_id"}, x3 sorted by sim]} and returns
{"mode": "answer" | "discriminate" | "name",
 "cards": [card strings to show, <=3],      # for mode answer; you may rewrite/compress a
     card text (e.g. keep only lines relevant to the question)
 "entity_ids": [ids of the entities those cards came from, same order],   # optional; lets
     the round report attribute outcomes to the right card
 "candidates": [names, <=3],                # for mode discriminate (their cards are looked
     up by name)
 "note": "optional extra text shown after the cards"}
Degrees of freedom left to the student: mode answer = 0 decisions; discriminate = 1 (choose
     among <=3); name = 1 (generate a name). The student profile below tells you which of
     these this student can execute.
```

```
rebuild_store: to regenerate the memory cards, add  "rebuild_store": {"instructions": "<how
     each card should be written; e.g. what facts to prioritise, length limit>"}  and set "
     memory.store": "v2". The runtime will have YOU (the teacher, offline) write one card per
      entity from the entity name alone, following your instructions; the student never sees
     anything else. Do this only if you believe richer cards help this student (the profile
     reports how much injected text it can absorb). The store can be rebuilt ONCE per build (
     the instructions you give first are the ones that count; later rebuild requests are
     ignored and the report will say so).

Constraints: the harness must run with zero large-model calls at inference; keep prompts
     short (the student is sensitive to extra text); subtractive edits (k=0, no system text,
     fewer modes) are welcome and the bare-student score is always reported so you can see
     whether the harness is helping at all.

Reply with ONE JSON object for the harness at the top level (the keys listed above directly
     in the object  -  do not wrap it in "harness"/"spec"; unknown keys are ignored). You may
      precede it with brief reasoning. Python hook sources must be JSON string values (escape
      newlines as \n).

## Task
InfoSeek (knowledge-intensive visual question answering). Each item is a photo of an entity (
     landmark, building, species, artwork, vehicle, food...) and a question about a fact of
     that entity (dates, dimensions, locations, taxonomy, counts, materials...). The student
     must identify the entity and recall or read the fact. Scoring: official InfoSeek relaxed
      exact match; numeric answers must fall within the gold range. The practice split has
     labels; the teacher (you) sees practice traces only. The test split is evaluated once at
      the end and is never shown to you.

## Student profile (measured on practice data; feed into your design)
(no profile measured)

Full profile JSON:
{
 "note": "no student profile available for this build"
}

## Store examples (3 of 1,380 cards)
<three cards of the current store, e.g.>
- Q1005204: <card>
- ...: <card>
- ...: <card>

## Bare student accuracy on the 600-item probe set (no harness): <bare probe accuracy, e.g.
     19.33>

Write the round-0 harness now.
```

Prompt 2: Teacher reflect prompt for InfoSeek (rounds 1–3): the same documentation, the round history, the current harness JSON, the traced report of the last round, and the instruction; followed by the head of the real round-0 report of the main-table build (two of its thirty stratified failure traces).

```
You are compiling an INFERENCE-TIME HARNESS for a small frozen vision-language model (the
     student). The harness is a JSON object; the runtime executes it. The student is called
     with an image and a text prompt and must answer a knowledge question about the entity in
      the image (InfoSeek: e.g. "What is the elevation of this mountain?", "Which family does
      this plant belong to?"). Answers are short strings or numbers; scoring is the official
     InfoSeek relaxed exact match (numbers must fall in the gold range).

At inference NO large model is available. The only deterministic signals the runtime can
     compute per item are:
- clip: the top-3 entity names from a 1,380-entity store (the gold entity is always in the
     store), each with a CLIP ViT-B-32 image->name cosine similarity (typically 0.2-0.4; top
     -1 is the gold entity about 52% of the time, top-3 contains it 68%), and that entity's
     memory card text.
- qtype: "num" if the question expects a number, else "str"; split: "unseen_question" or "
     unseen_entity" (InfoSeek's two test conditions; no other meaning here).
- The question text.
Cards: the store's current cards are mostly just the entity NAME (median 17 characters); only
      138 of 1,380 are full fact cards. You may request a regenerated store ("memory.store":
     "v2", written by you offline with your own instructions; its cost is charged to this
     build)  -  see "rebuild_store" below.

HARNESS JSON (all keys optional; defaults shown):
{
 "name": "short id", "rationale": "why this harness, 1-3 sentences",
 "prompt": {
```

```
   "format": "plain",                  # or "typed_fields": the DSPy-adapter serialisation (
     system message declaring typed input/output fields, [[ ## field ## ]] markers, answer
     field + completed marker; answer_suffix becomes the objective sentence; the output is
     parsed from the answer field unless parse_py is set). It is a fixed contract, not an
     optimiser; on DocVQA it was worth +8 bare / +3 with OCR for Gemma.
   "system": null,                     # a system message; the student profile says standing
     advice blocks HURT this student
   "before_question": "",              # text placed before the question
   "after_question": "",               # text placed after the question / cards
   "card_frame": "Retrieved from your memory store (may or may not describe the right entity)
     :\n{cards}",   # must contain {cards}
   "answer_suffix": "Answer the question with a single word, number, or short phrase."
 },
 "memory": {"k": 1, "sim_floor": 0.0, "max_card_chars": 600, "store": "v1"},   # k in 0..3 =
     how many CLIP top cards are shown by default (subject to the floor)
 "decode": {"max_new_tokens": 64},
 "discriminate_prompt": "Which of these is the entity shown in the image?\n{options}\n0. None
     of them\nAnswer with the number only.",   # used when a router returns mode "
     discriminate" (student picks 1..n or 0 = no card); must contain {options}
 "name_prompt": "What is the specific name of the entity ... Answer with the name only.",   #
     used when a router returns mode "name" (student names the entity; MiniLM links the name
     into the store; the profile says this student names correctly ~30% of the time)
 "name_sim_floor": 0.5,
 "router_py": null,    # OPTIONAL python source defining  def select(item) -> dict  ; runs in
     a sandbox (stdlib only, no network, no files)
 "parse_py": null      # OPTIONAL python source defining  def parse(text) -> str   ; extracts
     the final answer from the student's raw output
}

router select(item) receives item = {"uid", "question", "qtype", "split", "clip": [{"name","
     sim","card","entity_id"}, x3 sorted by sim]} and returns
{"mode": "answer" | "discriminate" | "name",
 "cards": [card strings to show, <=3],      # for mode answer; you may rewrite/compress a
     card text (e.g. keep only lines relevant to the question)
 "entity_ids": [ids of the entities those cards came from, same order],   # optional; lets
     the round report attribute outcomes to the right card
 "candidates": [names, <=3],                # for mode discriminate (their cards are looked
     up by name)
 "note": "optional extra text shown after the cards"}
Degrees of freedom left to the student: mode answer = 0 decisions; discriminate = 1 (choose
     among <=3); name = 1 (generate a name). The student profile below tells you which of
     these this student can execute.

rebuild_store: to regenerate the memory cards, add  "rebuild_store": {"instructions": "<how
     each card should be written; e.g. what facts to prioritise, length limit>"}  and set "
     memory.store": "v2". The runtime will have YOU (the teacher, offline) write one card per
      entity from the entity name alone, following your instructions; the student never sees
     anything else. Do this only if you believe richer cards help this student (the profile
     reports how much injected text it can absorb). The store can be rebuilt ONCE per build (
     the instructions you give first are the ones that count; later rebuild requests are
     ignored and the report will say so).

Constraints: the harness must run with zero large-model calls at inference; keep prompts
     short (the student is sensitive to extra text); subtractive edits (k=0, no system text,
     fewer modes) are welcome and the bare-student score is always reported so you can see
     whether the harness is helping at all.

Reply with ONE JSON object for the harness at the top level (the keys listed above directly
     in the object  -  do not wrap it in "harness"/"spec"; unknown keys are ignored). You may
      precede it with brief reasoning. Python hook sources must be JSON string values (escape
      newlines as \n).

You are in round <r> of the harness-distillation loop. History so far (bare student = 19.33
     on the probe set; best validated harness so far = 'v2_top1_ambig_discrim' at validation
     41.33):
- round 0: 'v2_top1_ambig_discrim' probe 36.0 | validation 41.33 | ACCEPTED (new best)

## The harness that produced this report (edit THIS object; keep the exact schema and key
     names)
{
 "name": "<the harness JSON that produced the report, in full>"
}

Below is the report of the LAST round's harness on the 600-item probe set, with raw student
     traces.

<ROUND REPORT: the traced report of the last round, see the excerpt below>

Decide the next harness. You may keep, edit, or simplify the current one; removing components
      is as legitimate as adding them. If you first want to see more traces, reply ONLY with
     {"query": {...}} using keys among correct (0/1), linked (none/correct/wrong), qtype (num
```

```
     /str), split, mode, sim_min, sim_max, contains  -  you get one query per round, up to 30
      traces. Otherwise reply with the complete new harness JSON (the full object, not a diff
     ), starting with a 'rationale' that names what in the report motivated each change.

================================================================================
EXCERPT OF A REAL ROUND REPORT (round 0 of the main-table InfoSeek build; 2 of its 30 failure
      traces shown)
================================================================================
### Round result for harness 'v2_top1_ambig_discrim'
Probe accuracy: 36.0 (n=150); bare student on the same items: 19.33
Fired (card shown): 88.7% | unparsed: 0.0% | mean output chars: 12.9 | student seconds: 17.8
Accuracy by DoF condition (linked = whether the card shown was the gold entity's; computed
     from practice labels, never shown to the student): {"correct": {"acc": 46.39, "n": 97},
     "none": {"acc": 35.29, "n": 17}, "wrong": {"acc": 8.33, "n": 36}}
By split/qtype: {"unseen_entity/num": {"acc": 31.25, "n": 16}, "unseen_entity/str": {"acc":
     44.07, "n": 59}, "unseen_question/num": {"acc": 31.25, "n": 16}, "unseen_question/str":
     {"acc": 30.51, "n": 59}}
By mode: {"answer": {"acc": 35.71, "n": 42}, "discriminate": {"acc": 36.11, "n": 108}}
Discriminate pass: {"n": 108, "picked_none": 15.7, "linked_correct_when_picked": 67.0}

#### 30 failure traces (stratified by linked condition; 96 failures total)
- uid=infoseek_val_00043574 split=unseen_entity qtype=num mode=answer linked=correct correct
     =0
  Q: What is the highest observed lifespan of this animal?
  gold: 27
  CLIP top-3 (name, sim): [('Sea otter', 0.318), ('Beaver', 0.266), ('Muskrat', 0.262)]
  cards shown: ['Entity: Sea otter\nScientific name: Enhydra lutris.\nType: Marine mammal.\
     nFamily: Mustelidae.\nOrder: Carnivora.\nNative range: Northern and eastern North
     Pacific c']
  student output: '15 years' -> parsed: '15 years'

- uid=infoseek_val_00072264 split=unseen_entity qtype=str mode=discriminate linked=correct
     correct=0
  Q: What is the vertex figure of this object?
  gold: line segment
  CLIP top-3 (name, sim): [('Hexagon', 0.282), ('Jigsaw puzzle', 0.267), ('Cornettes de Bise
     ', 0.251)]
  cards shown: ['Entity: Hexagon\nType: Polygon in Euclidean geometry.\nDefinition: Six-sided
      polygon with six vertices and six interior angles.\nRegular hexagon: All sides and ang
     ']
  candidates: ['Hexagon', 'Jigsaw puzzle', 'Cornettes de Bise'] chosen=1 pass1='1'
  student output: 'Hexagon' -> parsed: 'Hexagon'

[... 28 more traces ...]
```

Prompt 3: Fact-card generation prompt (one call per batch of ten entities, from entity names only; the teacher never sees questions or answers), with the card-writing instructions the teacher put in its `rebuild_store` request for the store the main-table build reuses.

```
Write one memory card per entity below, from the entity name alone. Follow these instructions
      exactly:
<the teacher's own instructions, written in the harness JSON as rebuild_store.instructions;
     the ones used for the store of the main-table build:>
Write one compact encyclopedic fact card for the given entity name. Length limit: 900
     characters. Use plain text, one fact per line, with stable labels when possible. First
     line must be `Entity: <canonical name>`. Include aliases/common names if important.
     Prioritize facts commonly asked in InfoSeek: entity type/category; country/city/region/
     location; dates/years founded, built, opened, created, born/died, discovered, dissolved;
      creators such as architect, artist, author, designer, manufacturer, founder; exact
     numeric facts with units such as elevation, height, length, width, area, depth, diameter
     , population, capacity, floors, spans, mass, speed, counts; taxonomy for organisms
     including family, genus, order, class, native range, habitat, conservation status, size;
      for buildings/landmarks include style, materials, use, owner/operator, dimensions; for
     artworks include artist, date, medium, dimensions, collection/location, genre; for
     vehicles include maker, production years, class, engine/power, speed/range; for foods
     include origin, main ingredients, type. Prefer high-confidence widely attested facts;
     omit uncertain facts rather than guessing. Keep wording terse and answer-like, e.g. `
     Family: Atherospermataceae.` Include both metric and imperial units only when commonly
     useful.

Entities:
<batch of 10 entities as 'Qid: entity name'>

Reply with ONLY a JSON object mapping each entity id to its card string, e.g. {"Q1": "card",
     "Q2": "card"}. Plain text cards, no markdown.
```

Prompt 4: One test item under the main-table InfoSeek harness: the router sends it to the bounded pick; pass 1 shows the three candidate names, pass 2 shows the chosen card in a fresh context; the parser extracts the answer.

```
ITEM infoseek_val_00000063 (test; image attached to every student call)
Question: Which country does this park come from?
Router decision: mode = discriminate (CLIP top-1 not confident), candidates = ["Huis Ten
     Bosch (theme park)", "Bristol Harbour", "Begijnhof, Amsterdam"]

--- PASS 1: the bounded pick (student sees the image + this text) ---
Which option is the entity shown in the image?
1. Huis Ten Bosch (theme park)
2. Bristol Harbour
3. Begijnhof, Amsterdam
0. None of them
If one option is plausible, choose it. Choose 0 only if every option is the wrong kind of
     thing. Answer with the number only.

Student output: 1   -> candidate 1 chosen; its card is shown in pass 2

--- PASS 2: the answer (student sees the image + this text; a fresh context) ---
Question:
Which country does this park come from?
Memory card for a possible entity:
- Entity: Huis Ten Bosch
Type: Theme park and resort.
Location: Sasebo, Nagasaki Prefecture, Japan.
Opened: 25 March 1992.
Theme: Dutch townscape, canals, gardens, windmills, and European-style architecture.
Area: About 152 ha (375 acres).
Name origin: Named after Huis ten Bosch palace in The Hague, Netherlands.
Operator: Huis Ten Bosch Co., Ltd.
Features: Seasonal flowers, illuminations, hotels, attractions, and harbor facilities.

Answer only the requested fact/value. Do not answer with the entity name unless the question
     asks for the name.

Student raw output: Japan
parse_py -> Japan   (gold: Japan; correct)

Items the router sends to mode 'answer' skip pass 1 and get the pass-2 prompt with the router
     's card; a pick of '0. None of them' answers without a card.
```

Prompt 5: The main-table InfoSeek harness (H_{val}, round 3 of the build under the round-0 restart rule; test 40.2): the JSON without its two code hooks.


```
{
 "name": "v2_type_terms_tasklines_discnote_rangeparse",
 "rationale": "The report showed that several breaks came from over-aggressive type filtering
      for buildings/monuments, so this version broadens building-like terms to keep the CLIP
     top entity when it is a memorial, monument, square, arch, column, fountain, or public
     artwork. Linked-correct failures also showed extraction misses for taxonomy, course type
     , parent peak, commemorates/style, launch complex, inflow, native/invaded range, and
     operator/managed-by facts, so the router now surfaces those lines more explicitly.
     Discrimination failures included the student choosing 0 despite a plausible candidate
     and answering order instead of genus/family for upper-taxonomy questions, so the
     discriminate prompt and notes are softened/clarified; parsing also handles approximate
     numbers and simple numeric ranges.",
 "prompt": {
  "format": "plain",
  "system": null,
  "before_question": "Question: ",
  "after_question": "",
  "card_frame": "Memory card for a possible entity:\n{cards}\n",
  "answer_suffix": "Answer only the requested fact/value. Do not answer with the entity name
     unless the question asks for the name."
 },
 "memory": {
  "k": 1,
  "sim_floor": 0.0,
  "max_card_chars": 900,
  "store": "v2"
 },
 "decode": {
  "max_new_tokens": 28,
  "samples": 1,
  "temperature": 0.0,
  "top_p": 1.0,
  "aggregate": "vote"
```

```
 },
 "discriminate_prompt": "Which option is the entity shown in the image?\n{options}\n0. None
     of them\nIf one option is plausible, choose it. Choose 0 only if every option is the
     wrong kind of thing. Answer with the number only.",
 "name_prompt": "What is the specific name of the entity shown in the image? Answer with the
     name only.",
 "name_sim_floor": 0.5
}
```

Prompt 6: The same harness's `router_py`: `select(item)` decides per item between showing a compressed card and the bounded pick, and rewrites the card toward the question's fact type.

```
def select(item):
    import re
    clips = item.get('clip') or []
    if not clips:
        return {'mode': 'answer', 'cards': []}
    q = (item.get('question') or '').lower()

    def sim(c):
        try:
            return float(c.get('sim', 0.0) or 0.0)
        except Exception:
            return 0.0

    def note_for_question():
        notes = []
        if 'closest parent taxonomy' in q or 'parent taxonomy' in q:
            notes.append('Hint: answer the immediate parent taxon; for a species this is
     usually the genus, the first word of the scientific name.')
        elif 'closest upper taxonomy' in q or 'upper taxonomy' in q:
            notes.append('Hint: answer the closest taxon asked for, usually genus or family;
     do not answer the order unless the question asks for order.')
        if 'vertex figure' in q:
            notes.append('Hint: answer the geometric vertex-figure term, not the shape name.'
     )
        if 'named after' in q or 'named for' in q or 'name after' in q:
            notes.append('Hint: answer the namesake, not the entity itself.')
        if 'which road, railway or canal' in q or 'does this bridge carry' in q or 'has this
     bridge crossed' in q or 'crossed over' in q:
            notes.append('Hint: answer only what is carried or crossed, not the bridge name.'
     )
        if 'who found' in q:
            notes.append('Hint: this means founder/founded by, not architect or designer.')
        if 'who designed' in q:
            notes.append('Hint: answer the designer/architect name only.')
        if 'commemorate' in q:
            notes.append('Hint: answer the event or subject commemorated.')
        if 'architectural style' in q:
            notes.append('Hint: answer the style/order only.')
        if item.get('qtype') == 'num':
            notes.append('Hint: give a number in the unit requested by the question.')
        return ' '.join(notes)

    def expected_terms():
        checks = []
        if 'plant' in q:
            checks += ['plant', 'tree', 'shrub', 'flower', 'grass', 'conifer', 'botanical', '
     herb', 'orchid']
        if 'bird' in q:
            checks += ['bird', 'aves', 'songbird', 'waterfowl', 'seabird', 'wading bird']
        if 'fish' in q:
            checks += ['fish']
        if 'insect' in q:
            checks += ['insect', 'beetle', 'moth', 'butterfly', 'dragonfly']
        if 'animal' in q:
            checks += ['animal', 'mammal', 'bird', 'fish', 'reptile', 'insect', 'butterfly',
     'dragonfly', 'canid', 'felid', 'cat', 'dog']
        if 'bridge' in q:
            checks += ['bridge', 'viaduct']
        if 'building' in q:
            checks += ['building', 'church', 'cathedral', 'castle', 'basilica', 'mosque', '
     temple', 'museum', 'hall', 'synagogue', 'tower', 'palace', 'lighthouse', 'memorial', '
     monument', 'arch', 'column', 'fountain', 'square', 'plaza', 'market', 'cemetery', '
     fortress', 'sculpture', 'public art']
        if 'aircraft' in q:
            checks += ['aircraft', 'airliner', 'airplane', 'aeroplane', 'fighter', '
     manufacturer', 'twinjet']
        if 'lake' in q:
```

```
            checks += ['lake', 'reservoir', 'mere']
        if 'mountain' in q or 'peak' in q:
            checks += ['mountain', 'peak', 'summit', 'massif', 'range', 'pass']
        if 'drink' in q:
            checks += ['drink', 'beverage', 'cocktail', 'coffee']
        if 'food' in q:
            checks += ['food', 'dish', 'cuisine', 'ingredient', 'cocktail', 'pasta']
        if 'material' in q or 'crystal system' in q or 'decomposition point' in q:
            checks += ['material', 'compound', 'chemical', 'mineral', 'polymer', 'crystal']
        return checks

    def type_matches(c, terms):
        if not terms:
            return True
        txt = ((c.get('name') or '') + '\n' + (c.get('card') or '')).lower()
        for t in terms:
            if t in txt:
                return True
        return False

    def val_after_label(lines, labels):
        for lab in labels:
            llab = lab.lower()
            for ln in lines:
                low = ln.lower()
                if low.startswith(llab + ':'):
                    return ln.split(':', 1)[1].strip().strip('.')
        return ''

    def genus_from_lines(lines):
        g = val_after_label(lines, ['Genus'])
        if g:
            return g.split(';')[0].split(',')[0].strip()
        sci = val_after_label(lines, ['Scientific name', 'Binomial name'])
        m = re.match(r'([A-Z][A-Za-z\-]+)\s+[a-z]', sci)
        if m:
            return m.group(1)
        return ''

    def relevant_card(c):
        name = c.get('name') or ''
        card = c.get('card') or name
        raw_lines = [ln.strip() for ln in card.splitlines() if ln.strip()]
        body = []
        for ln in raw_lines:
            if not ln.lower().startswith('entity:'):
                body.append(ln)
        task_lines = []
        gen = genus_from_lines(body)
        fam = val_after_label(body, ['Family'])
        order = val_after_label(body, ['Order'])
        if ('closest parent taxonomy' in q or 'parent taxonomy' in q) and gen:
            task_lines.append('Requested parent taxonomy: ' + gen + '.')
        elif ('closest upper taxonomy' in q or 'upper taxonomy' in q):
            if gen:
                task_lines.append('Possible closest upper taxonomy/genus: ' + gen + '.')
            if fam:
                task_lines.append('Family: ' + fam + '.')
        elif 'family' in q and fam:
            task_lines.append('Requested family: ' + fam + '.')
        elif 'order' in q and order:
            task_lines.append('Requested order: ' + order + '.')
        elif 'genus' in q and gen:
            task_lines.append('Requested genus: ' + gen + '.')

        def add_value(label, labels):
            v = val_after_label(body, labels)
            if v:
                task_lines.append(label + ': ' + v + '.')
        if 'parent peak' in q:
            add_value('Requested parent peak', ['Parent peak', 'Parent mountain'])
        if 'course type' in q or ('course' in q and 'food' in q):
            add_value('Requested course type', ['Course', 'Course type', 'Type'])
        if 'commemorate' in q:
            add_value('Commemorates', ['Commemorates', 'Commemorated event', 'Dedicated to'])
        if 'architectural style' in q or 'style of this building' in q:
            add_value('Architectural style', ['Architectural style', 'Style', 'Order'])
        if 'street address' in q or 'address' in q:
            add_value('Address', ['Address', 'Street address', 'Location'])
        if 'city or region' in q or 'locate in' in q:
            add_value('City/region', ['City', 'Region', 'Location', 'Administrative area'])
```

```
        if 'country' in q and 'country' not in ''.join(task_lines).lower():
            add_value('Country', ['Country', 'Location'])
        if 'continent' in q:
            add_value('Continent', ['Continent', 'Location'])
        if 'native' in q or 'native to' in q:
            add_value('Native range', ['Native range', 'Native to', 'Range', 'Distribution'])
        if 'invaded' in q or 'invasive' in q:
            add_value('Invaded location', ['Invasive in', 'Introduced to', 'Invaded range', '
    Range'])
        if 'inflow' in q:
            add_value('Inflow', ['Inflow', 'Primary inflows', 'Inflow source'])
        if 'starting point' in q or 'start' in q or 'launch' in q:
            add_value('Launch/starting point', ['Launch complex', 'Launch site', 'Launched
    from', 'Starting point', 'Origin'])
        if 'operates' in q or 'operator' in q or 'operated' in q or 'managed' in q:
            add_value('Operator/manager', ['Operator', 'Operated by', 'Managed by', '
    Administered by', 'Governing body'])
        if 'data transfer' in q or 'logical block addressing' in q:
            add_value('Data transfer speed', ['Data rate', 'Bit rate', 'Transfer rate', '
    Speed'])

        keys = []
        def add(*xs):
            for x in xs:
                keys.append(x)
        if 'family' in q:
            add('family')
        if 'order' in q:
            add('order')
        if 'genus' in q or 'closest parent taxonomy' in q or 'parent taxonomy' in q or '
    closest upper taxonomy' in q or 'upper taxonomy' in q:
            add('genus', 'scientific name', 'binomial', 'family', 'order')
        if 'parent peak' in q:
            add('parent peak', 'parent mountain')
        if 'native' in q or 'originally discovered' in q or 'where is' in q or 'locate' in q
    or 'located' in q or 'city' in q or 'region' in q or 'country' in q or 'continent' in q:
            add('location', 'located', 'city', 'region', 'country', 'continent', 'native
    range', 'range', 'origin', 'discovered')
        if 'invaded' in q or 'invasive' in q:
            add('invasive', 'introduced', 'invaded', 'feral')
        if 'who found' in q or 'founded' in q or 'founder' in q:
            add('founder', 'founded by', 'founded', 'established by', 'benefactor', 'patron')
        if 'designed' in q or 'designer' in q or 'architect' in q:
            add('designed by', 'designer', 'architect', 'engineer')
        if 'opened' in q or 'built' in q or 'created' in q or 'published' in q or 'first
    mentioned' in q or 'year' in q or 'retired' in q or 'operational service' in q:
            add('opened', 'built', 'created', 'published', 'first mentioned', 'founded', '
    date', 'year', 'retired', 'withdrawn', 'introduced')
        if 'named after' in q or 'named for' in q:
            add('named for', 'named after', 'namesake', 'etymology', 'name origin')
        if 'operates' in q or 'operator' in q or 'operated' in q or 'managed' in q:
            add('operator', 'operated by', 'managed by', 'administered by', 'governing body')
        if 'carr' in q:
            add('carries', 'type')
        if 'cross' in q:
            add('crosses', 'crossed', 'river')
        if 'lifespan' in q or 'life span' in q:
            add('lifespan', 'longevity', 'maximum age')
        if 'offspring' in q or 'at the same time' in q or 'eggs' in q:
            add('offspring', 'clutch', 'litter', 'brood', 'eggs')
        if 'velocity' in q or 'speed' in q or 'kilometre per hour' in q or 'km' in q or 'data
     transfer' in q or 'bit per second' in q or 'logical block addressing' in q:
            add('speed', 'velocity', 'km/h', 'bit/s', 'kbit/s', 'mbit/s', 'data rate', 'bit
    rate', 'transfer rate')
        if 'frequency' in q or 'sound' in q or 'hear' in q:
            add('frequency', 'hearing', 'sound', 'hz')
        if 'weight' in q or 'mass' in q:
            add('weight', 'mass')
        if 'height' in q or 'elevation' in q or 'sea level' in q or 'length' in q or 'width'
    in q or 'area' in q or 'density' in q or 'decomposition' in q or 'depth' in q or '
    wingspan' in q:
            add('height', 'elevation', 'sea level', 'length', 'width', 'wingspan', 'area', '
    density', 'decomposition', 'depth', 'point')
        if 'starting point' in q or 'start' in q or 'launch' in q:
            add('starting point', 'launch site', 'launch complex', 'launched from', 'origin')
        if 'genre' in q:
            add('genre', 'type', 'category', 'public art')
        if 'historic county' in q:
            add('historic county', 'county', 'location')
        if 'crystal system' in q:
            add('crystal system')
```

```
        if 'facet' in q:
            add('facet', 'side', 'face')
        if 'course type' in q:
            add('course', 'course type', 'served')
        if 'commemorate' in q:
            add('commemorates', 'commemorated', 'dedicated')
        if 'architectural style' in q or 'style of this building' in q:
            add('architectural style', 'style', 'order')
        if 'inflow' in q:
            add('inflow', 'primary inflows')

        scored = []
        for i, ln in enumerate(body):
            low = ln.lower()
            score = 0
            for k in keys:
                if k in low:
                    score += 5
            for w in q.replace('?', ' ').replace(',', ' ').replace('(', ' ').replace(')', ' '
    ).split():
                if len(w) > 4 and w in low:
                    score += 1
            scored.append((-score, i, ln))
        chosen = [ln for sc, i, ln in sorted(scored) if sc < 0]
        if len(chosen) < 6:
            for ln in body:
                if ln not in chosen:
                    chosen.append(ln)
                if len(chosen) >= 10:
                    break
        else:
            chosen = chosen[:10]
        head = 'Likely entity, not usually the answer: ' + name if name else 'Possible memory
     facts:'
        out_lines = [head] + task_lines + chosen
        out = '\n'.join(out_lines)
        return out[:900]

    top = clips[0]
    s1 = sim(top)
    s2 = sim(clips[1]) if len(clips) > 1 else -1.0
    s3 = sim(clips[2]) if len(clips) > 2 else -1.0
    gap12 = s1 - s2
    gap13 = s1 - s3
    note = note_for_question()
    terms = expected_terms()

    if terms and s1 < 0.335 and not any(type_matches(c, terms) for c in clips[:3]):
        return {'mode': 'answer', 'cards': [], 'note': note}

    cands = [c for c in clips[:3] if (c.get('name') and sim(c) >= 0.17)]
    if terms:
        matching = [c for c in cands if type_matches(c, terms)]
        if matching:
            cands = matching

    if ('data transfer' in q or 'logical block addressing' in q) and cands:
        txt = ((cands[0].get('name') or '') + '\n' + (cands[0].get('card') or '')).lower()
        if 'compact disc' in txt or 'cd-da' in txt or 'audio cd' in txt:
            return {'mode': 'answer', 'cards': [relevant_card(cands[0])], 'entity_ids': [
     cands[0].get('entity_id')], 'note': note}

    ambiguous = False
    if len(cands) >= 2 and gap12 < 0.035 and s1 < 0.34:
        ambiguous = True
    if len(cands) >= 3 and gap13 < 0.060 and s1 < 0.30:
        ambiguous = True
    if len(cands) >= 2 and s1 < 0.235:
        ambiguous = True

    if ambiguous:
        return {'mode': 'discriminate', 'candidates': [c.get('name') for c in cands[:3]], '
     note': note}

    if terms and not type_matches(top, terms) and cands and type_matches(cands[0], terms):
        top = cands[0]

    return {'mode': 'answer', 'cards': [relevant_card(top)], 'entity_ids': [top.get('
     entity_id')], 'note': note}
```

Prompt 7: The same harness's `parse_py`: extracts the final short answer from the student's raw output.

```
def parse(text):
    import re
    s = '' if text is None else str(text)
    s = s.strip()
    m = re.search(r'\[\[\s*##\s*answer\s*##\s*\]\](.*?)(?:\[\[\s*##|\Z)', s, re.I | re.S)
    if m:
        s = m.group(1).strip()
    s = re.sub(r'\[\[\s*##\s*completed\s*##\s*\]\]', '', s, flags=re.I).strip()
    parts = re.split(r'(?i)\banswer\s*[:\-]\s*', s)
    if len(parts) > 1:
        s = parts[-1].strip()
    s = re.sub(r'(?is)^(the\s+answer\s+is|it\s+is|it\'s)\s+', '', s).strip()
    lines = [ln.strip() for ln in s.splitlines() if ln.strip()]
    if lines:
        s = lines[0]
    s = re.sub(r'[*_]+', '', s)
    s = s.strip(' "\'`')
    s = re.sub(r'\s+', ' ', s)
    s = re.sub(r'[\..]+$', '', s).strip()
    s = re.sub(r'(?i)^(about|approximately|approx\.?|around|roughly|ca\.?|c\.)\s+', '', s).
     strip()

    def fmt_num(x):
        try:
            if abs(x - round(x)) < 1e-9:
                return str(int(round(x)))
            out = ('%.10f' % x).rstrip('0').rstrip('.')
            return out
        except Exception:
            return str(x)

    # Simple numeric ranges: return midpoint, useful for clutch sizes/lifespans when gold
     accepts a relaxed numeric range.
    m = re.match(r'^([-+]?\d[\d,]*(?:\.\d+)?)\s*(?:-|\u2013|\u2014|to)\s*([-+]?\d[\d,]*(?:\.\
     d+)?)(?:\s*(eggs?|offspring|years?|yr|g|grams?|kg|cm|mm|m|metres?|meters?|km|hz|khz|mhz)
     )?\.?$', s, re.I)
    if m:
        a = float(m.group(1).replace(',', ''))
        b = float(m.group(2).replace(',', ''))
        unit = (m.group(3) or '').lower()
        if unit or max(abs(a), abs(b)) <= 1000:
            return fmt_num((a + b) / 2.0)

    # Normalize common data-rate units into bit/s when the model gives equivalent units.
    m = re.match(r'^([-+]?\d[\d,]*(?:\.\d+)?)\s*(kbit/s|kb/s|kbps|kilobits? per second)\b', s
     , re.I)
    if m:
        return fmt_num(float(m.group(1).replace(',', '')) * 1000.0)
    m = re.match(r'^([-+]?\d[\d,]*(?:\.\d+)?)\s*(mbit/s|mb/s|mbps|megabits? per second)\b', s
     , re.I)
    if m:
        return fmt_num(float(m.group(1).replace(',', '')) * 1000000.0)
    m = re.match(r'^([-+]?\d[\d,]*(?:\.\d+)?)\s*(gbit/s|gb/s|gbps|gigabits? per second)\b', s
     , re.I)
    if m:
        return fmt_num(float(m.group(1).replace(',', '')) * 1000000000.0)
    m = re.match(r'^([-+]?\d[\d,]*(?:\.\d+)?)\s*(bit/s|bits? per second|bps)\b', s, re.I)
    if m:
        return fmt_num(float(m.group(1).replace(',', '')))

    # Pure numbers with thousands commas.
    if re.match(r'^[-+]?\d{1,3}(?:,\d{3})+(?:\.\d+)?$', s):
        return s.replace(',', '')

    # For numeric answers, remove common trailing units while preserving the value.
    m = re.match(r'^([-+]?\d+(?:\.\d+)?)(?:\s*(?:eggs?|offspring|years?|year|yr|km/h|
     kilometres? per hour|kilometers? per hour|m/s|hz|khz|mhz|grams?|grammes?|g|kg|kilograms
     ?|m|metres?|meters?|cm|mm|km|ft|feet|miles?|sq\.?\s*km|square kilometres?|square
     kilometers?|\u00b0?\s*c|degrees? celsius|%)\b)\.?$', s, re.I)
    if m:
        return m.group(1)
    return s
```

B COMPILER ABLATIONS AND BUILD STABILITY

B.1 TEACHER, PROFILE AND FEEDBACK ABLATIONS

Table 14: Compiler-input ablations. These earlier builds use the budgets in Appendix F.1.

Configuration	InfoSeek	DocVQA
DSL default: name card / plain OCR	23.6	80.1
Store-only / OCR-only control	33.5	80.1
Sensitivity profile (3 / 2 builds)	33.7 to 34.0	84.9 / 85.7
Profile withheld (3 / 1 builds)	35.6 / 39.7 / 40.3	86.0
Measurements-only profile	34.0	n/a
Decision-menu profile	38.8	n/a
Score-only report	30.2	72.9
Selected harness with router removed	n/a	85.1 / 86.4
Prompt-only loop	18.7	80.4
Student as teacher, text-only	23.6	85.0
gpt-5.4-mini / local 30B teacher	22.5 / 23.5	85.3 / 85.1

Traces diagnose failures that scores alone do not. With a score-only report, InfoSeek reaches 30.2 and DocVQA 72.9; the latter build removes OCR after a parse failure it cannot diagnose. Traced feedback repairs broken parse hooks and identifies harmful components. These ablations support using execution evidence, while the one-shot results below show that reflection is not necessary on every task.

Profile content matters. Sensitivity profiles produce InfoSeek builds at 33.7–34.0 that retain no student selection. No-profile builds keep a bounded pick, while a decision-menu profile also retains it (38.8; round 0: 38.15). The comparisons involve a small number of builds and different budgets; they support a distinction between profile formats, not a universal benefit from withholding measurements.

Teacher capacity is task-dependent. On DocVQA, GPT-5.5, gpt-5.4-mini, the local 30B teacher and the student itself reach similar scores. On InfoSeek, the smaller teachers do not elect to regenerate the fact store and remain near the name-card default. Teacher-written content and feedback-driven control are therefore separate sources of improvement.

B.2 ONE-SHOT CONSTRUCTION

A second reference point for the recipes above: a harness the teacher writes in one GPT-5.5 call per bed (USD 1.44 across the original eight-setting batch) from the runtime's manual with every guidance sentence removed by a fixed rule (references to the student profile, degrees of freedom, calibration facts, and "prefer" / "should" sentences; 2–8 sentences per bed, logged), plus the task description; the harness is validated by the runtime and tested once. Tab. 15 pairs it against the compiled harness and the human recipe: performance varies with the interface and the decisions that must be fitted to the student. One call matches the selected build on depth and is statistically tied on reflectance and DocVQA; it falls below the compared loop on InfoSeek, correspondence, LiveVQA and SlideVQA. In particular, the InfoSeek one-shot result (25.0) is below the public-content human recipe (33.8). The manual supplies tool descriptions and an editable interface, so this comparison measures the value of iteration within that interface.

B.3 BUILD VARIATION AND VALIDATION SELECTION

The original DocVQA 150/300 build begins with a parse hook that discards answers (test ANLS 0.2 on the 0–100 scale); the first traced reflection repairs it. This is a failed end-to-end harness, not the bare model's reading score of 72.7. The later restart-rule build is a different run, with test $H_0 = 83.0$ and $H_{\text{val}} = 85.2$ (Appendix B.4); its initial score must not be paired with the original build's 85.6.

Table 15: Independent one-shot construction versus iterative HC. Reference builds are listed in Table 10.

Setting	One-shot	HC	One-shot minus HC
BLINK reflectance	65.7	64.9	$+0.7$ [-7.5, $+9.0$], n.s.
BLINK correspondence	62.8	67.6	-4.8 [-9.1, -0.7], $p = .035$
BLINK depth	95.2	95.2	Identical verdicts
DocVQA	85.5	84.9	$+0.6$ [-0.3, $+1.4$], n.s.
InfoSeek	25.0	39.7	-14.7 [-15.8, -13.5]
LiveVQA	30.6	37.1	-6.5, $p = 4 \times 10^{-14}$
SlideVQA	48.3	56.7	-8.4 [-10.2, -6.5]

Table 16: Initial and selected harnesses with paired statistical details. The DocVQA row retains the original build; Table 2 instead uses its restart-rule rerun, described below.

Setting	H_0	H_{val}	Gain over bare	Paired p
InfoSeek	36.2	40.2	$+22.2$	$< 10^{-250}$
DocVQA (original)	0.2	85.6	$+12.9$	$< 10^{-50}$
BLINK reflectance	59.7	64.9	$+14.2$	.015
BLINK correspondence	66.7	67.6	$+12.7$	4×10^{-8}
BLINK depth	95.2	95.2	$+12.9$	4×10^{-4}
LiveVQA (judged)	28.1	37.1	$+22.4$	3×10^{-86}
SlideVQA	41.9	56.7	$+12.0$	2×10^{-27}

Reflection improves the selected InfoSeek restart-rule build from 36.2 to 40.2, LiveVQA from 28.1 to 37.1 and SlideVQA from 41.9 to 56.7. Depth keeps round 0 in all six folds across two seeds. Validation selection can nevertheless choose a round that is worse on held-out test items (Fig. 3).

B.4 Failed initial builds and the restart safeguard

Before the restart safeguard, two InfoSeek builds began with compact-card routers that filtered out useful fact lines. A keep-better gate could accept a poor first candidate because it compared candidates only with each other. The failed 8B build is retained alongside its successful second seed. The 4B reflections recover to 34.1. The 8B reflections instead remove the store on 93% of test items, reaching 23.2 against 34.7 for the transplanted harness. Its second seed reaches 40.3 (Table 22); both outcomes matter when assessing recompilation.

Restart rule and its evidence. If round 0 scores below bare on the probe, the teacher receives a traced diagnosis and may recompile it at most twice before validation selection; bare also remains a validation candidate. The updated InfoSeek build uses the same seed, splits and store as the original 150/300 build, but its first draw already clears the probe floor, so no restart is executed. It reaches 40.2, versus the original 34.1; this difference is evidence of build variability, not an isolated causal

Table 17: Two original InfoSeek builds with below-bare probe scores at round 0. Entries are probe / validation accuracy; a check marks an accepted round.

Stage	Qwen3-VL-4B, seed 0	Qwen3-VL-8B, seed 0
Bare	19.3 / 20.3	22.7 / 22.0
Round 0	17.3 / 22.3 ✓	10.0 / 15.3 ✓
Round 1	31.3 / 37.7 ✓	24.7 / 23.7 ✓
Round 2	32.7 / 36.3	25.3 / 25.7 ✓
Round 3	32.0 / 33.7	28.7 / 25.7
Test $H_0 \rightarrow H_{\text{val}}$	$22.9 \rightarrow 34.1$	$14.7 \rightarrow 23.2$
Test items shown a card	4,873 / 6,000	414 / 6,000

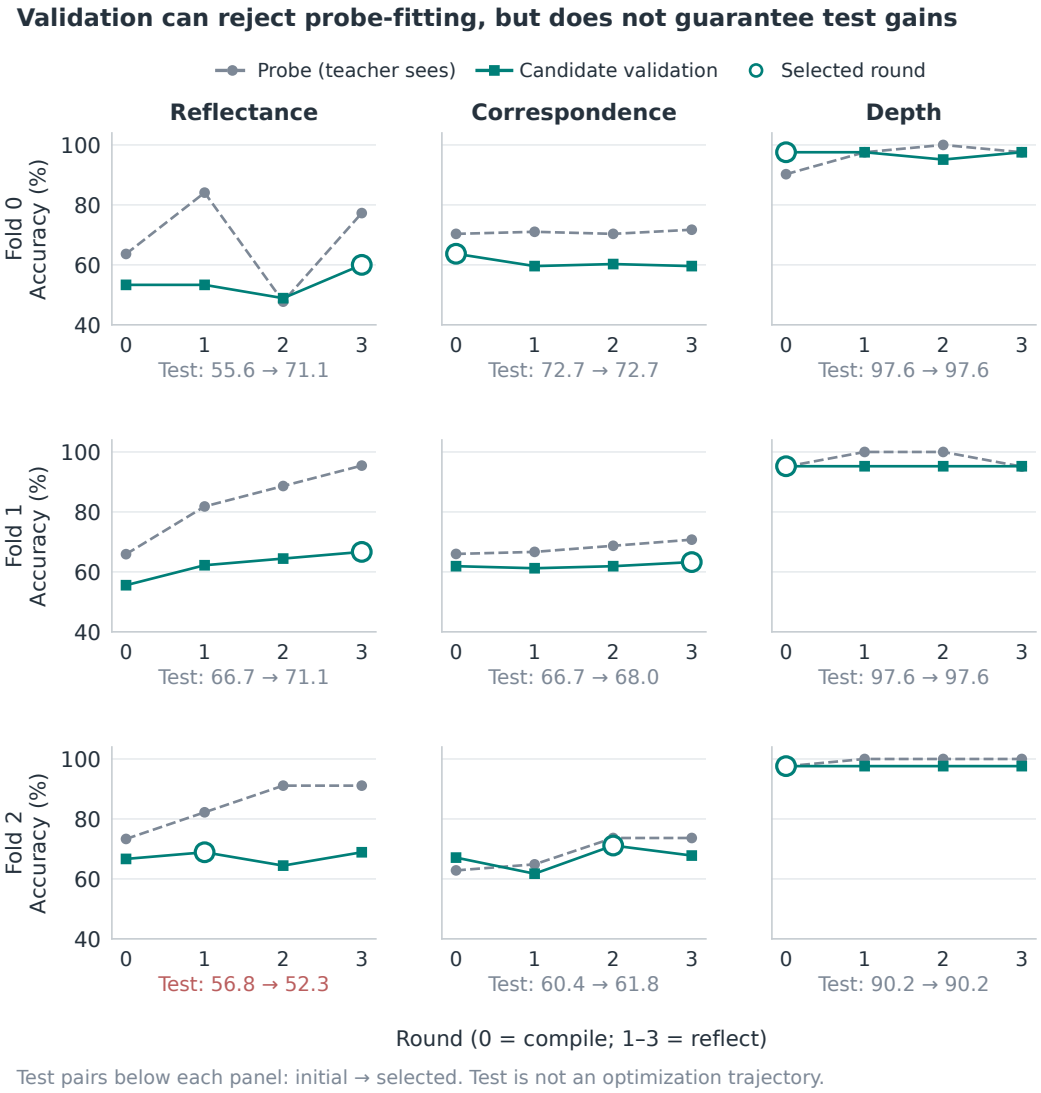


Figure 3: **Probe improvements can fail to generalize.** Seed-0 BLINK trajectories, with tasks in columns and folds in rows. Dashed: teacher-visible probe; solid: candidate validation score; ring: gate-selected round. Held-out test scores are printed below each panel as initial → selected, not used for selection. Reflectance fold 2 loses test accuracy despite its gate improvement. These are the archived per-fold trajectories replotted at readable scale.

effect of restarting. On DocVQA the safeguard does fire: the first candidate scores 50.8 on the probe against bare 68.3, the replacement scores 80.3, and the selected build reaches 85.2 on test against 85.6 for the earlier build. Older comparisons retain their original build scores and are identified as such.

B.5 Execution feedback and decision allocation

The teacher receives stage-specific diagnostics and a stratified sample of failure traces, then writes a revised harness with a rationale linking its edits to the report. Table 18 follows this process in three InfoSeek builds. The reported percentages describe each candidate's execution on its 150-item probe. They are not test scores and are not used in place of the separate validation gate.

Table 18: **Execution feedback changes the student's role.** Probe = accuracy; Pick = percentage of all probe items routed to a candidate pick; Abstain = percentage of those picks choosing "none". Shares are rounded. A dash means no pick was invoked.

Round	Probe	Pick	Abstain	Reported failure and resulting revision
Qwen3-VL-8B, seed 1: add a choice, then bound it				
0	36.7	0%	–	Initial harness shows two or three cards without an explicit pick.
1	37.3	87%	11%	Wrong-card failures: add a bounded student pick on ambiguous links.
2	39.3	84%	25%	Pick overturns correct top-1 links: narrow to two candidates and favor top-1.
3	48.0	77%	16%	High-confidence matches: bypass the pick; prioritize relevant fact lines.
Qwen3-VL-4B, main build: revise activation and the answer interface				
0	36.0	72%	16%	Initial harness defaults to top-1 and invokes a pick on ambiguous links.
1	38.0	69%	18%	Wrong cards and entity-name answers: filter incompatible cards and request the fact explicitly.
2	40.0	61%	18%	Unit parsing and taxonomy failures: repair extraction and prepend a task-value line.
3	42.7	62%	11%	Overactive filtering and "none" despite plausible options: relax the filter and soften the pick prompt.
Qwen3.5-2B: retain selection but remove its abstention option				
0	23.3	67%	76%	Initial bounded pick includes "none".
1	28.7	86%	0%	Frequent abstention despite useful picks: remove "none", retaining the candidate choice.

These trajectories show that feedback can add a student decision, restrict its options or move cases back to deterministic routing. Reordering card lines and repairing parsers can also improve the interface without changing who makes the final choice. Multiple components change within a round, and correct-card, wrong-card and no-card groups contain different items. Neither their score differences nor successive round scores isolate the causal effect of an individual edit. The controlled interface interventions in Table 4 provide the separate evidence for those effects.

C Decision diagnostics and cross-student transfer

C.1 Student-routed and single-decision controls

The student-routed control exposes the same resources with a plain instruction: three candidate cards on InfoSeek and LiveVQA, page OCR on DocVQA, rounded raw signals on BLINK, and the whole deck plus OCR of five ranked slides on SlideVQA. It removes compiled routing, verdicts, parsers and output contracts. Main scores are in Table 1; Table 19 retains the paired uncertainty rather than repeating all baseline columns.

One interface at a time. Table 4 reports the main-build score changes. Table 20 specifies the operational change in each column. The 4B requests a card on 58% of InfoSeek items and slides on 71% of SlideVQA items. Gemma requests OCR on 5% of DocVQA items and instead answers the

Table 19: Student-routed minus HC on the same test items. Intervals are reported 95% intervals; absent intervals are not estimated here.

Setting	Paired difference	Uncertainty / paired test
InfoSeek	-11.0	219 fixed / 880 broken; $p = 3 \times 10^{-94}$
DocVQA	-5.1	$[-6.2, -4.0]$
SlideVQA	-12.2	$[-14.3, -10.2]$
LiveVQA	-0.2	Not significant
BLINK reflectance	-14.9	$[-24.6, -5.2]$; $p = .006$
BLINK correspondence	-2.7	$[-6.6, +1.1]$; not significant
BLINK depth	-21.8	$[-30.6, -13.7]$

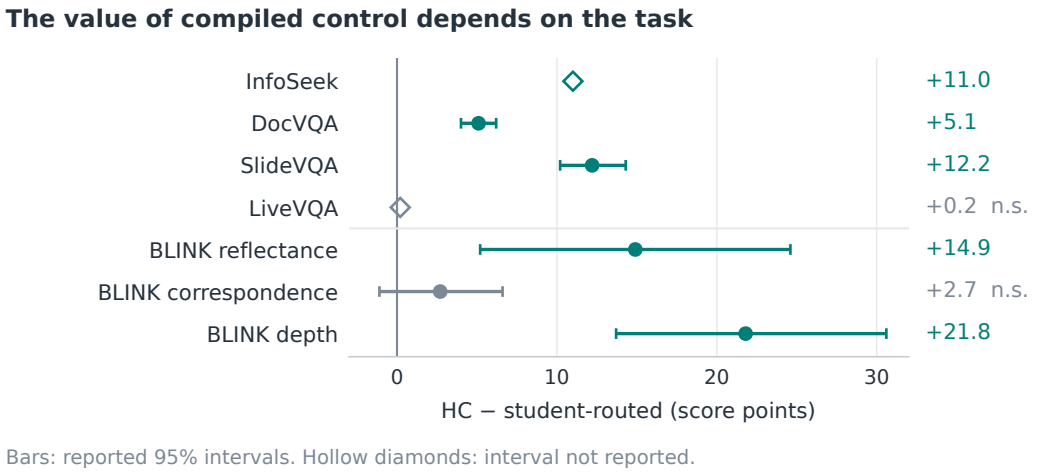


Figure 4: **Compiled-control gains vary by task.** HC minus student-routed test scores. Bars show reported 95% intervals. Hollow diamonds have no reported interval. Build identities are in Table 10.

underlying question on 92%. BLINK request rates range from 80% to 100%. These rates describe the tested contracts rather than general tool-use ability.

Table 20: Definitions of the decision-interface interventions.

Decision	Intervention	Interpretation
Decide	Supply the resource only after a yes/no request	Tests activation under that request contract
Select	Show all candidates without the explicit pick or ranker	Changes the selection interface, not only the decision owner
Generate	Use a student-generated entity name for lookup	Tests open argument generation on InfoSeek
Integrate	Use raw content instead of rewritten content. On BLINK, use raw signals instead of the processed rule output	The intervention differs between content and perception tasks
Persist	Remove the answer contract and parser. On BLINK, show the verdict as a suggestion	Tests answer and abstention control, or verdict emission

The interventions are not independent and their effects are not additive. On BLINK, the raw-signal integration arm is also the all-back arm. Each ablation is a single unselected variant compared with a validation-selected harness. Nonsignificance does not establish equivalence across all prompts. In particular, InfoSeek's selected harness already retains a student pick. Its select ablation removes the explicit two-pass choice and presents candidate content together. It should not be described simply as handing a code decision back to the student.

Table 21: InfoSeek interventions under two build versions. Entries are score changes.

HC reference	Decide	Select	Generate	Integrate	Persist	All back
Main, 40.2	-9.7	-2.2	-13.8	-0.3^{ns}	-5.5	-11.0
Original, 34.1	-7.5	-0.0^{ns}	-9.6	$+2.5$	-2.3	-4.9

The original build began below bare on the probe and predates the restart-rule procedure. Both sets of results are retained rather than treated as repeated measurements of the same harness. The all-back comparisons for the other tasks are recorded in Table 19.

C.2 Decision ownership and student-specific policies

Figures 5 and 6 describe the selected programs, not a common capability scale. Reading supplied text, selecting a candidate, and answering after a tool defers are different events. We therefore compare policies within the same task and omit averages over different mixtures of tasks and students. InfoSeek retains a bounded pick on ambiguous items; content tasks retain student reading, while BLINK rules answer 55–93% of items in the source builds. LiveVQA and SlideVQA leave similar broad roles to the student but have very different control gains, so the number of compiled decisions does not predict their value.

C.3 Cross-student transfer and recompilation

Two source harnesses serve distinct comparisons. Table 22 compares the original 450-item InfoSeek source build with own builds at that budget (including the source student's restart-rule rerun). Tables 23 and 24 diagnose the earlier 1,200-item discriminate harness and distinguish its rewritten store from the reference store. These columns must not be treated as repeated measurements of one identical harness. Additional students receive each source harness unchanged. Some diagnostic measurements were collected after transfer evaluation; the correlations characterize transfer behavior rather than constitute an independent prospective prediction test. The routed decision has distinct failure modes. Qwen3.5-2B takes the escape option on 85% of routed items; MiniCPM's linking precision is close to the linker's, leaving little selection benefit. Whole-harness gains also include answer-side edits and, where applicable, a changed store. The with-sensitivity-profile harness is within two points of store-only on the five students for which it was evaluated.

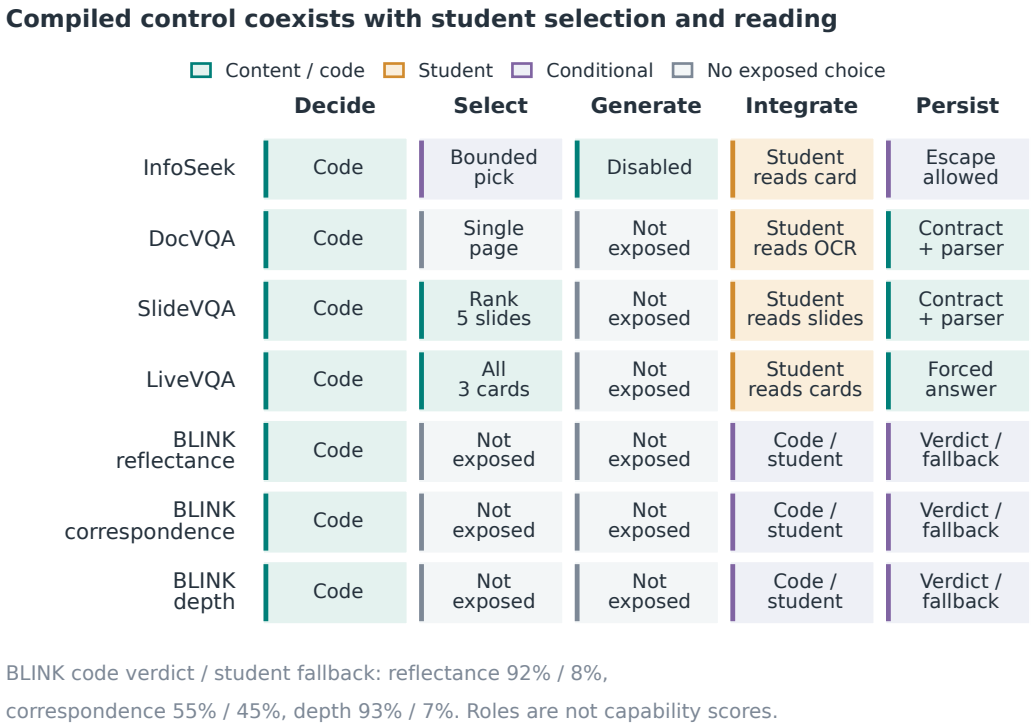


Figure 5: **Decision ownership is a property of the selected program and task interface.** Colors distinguish content/code, student work, conditional sharing and no exposed choice. This is a categorical description, not a capability heatmap: a single-page input and a compiled multi-page selector do not mean the same thing. BLINK percentages refer to code verdicts versus student fallback, not to content-reading rates. The VLA policy interface is separate (App. G.2).

Table 22: InfoSeek transfer and recompilation at 450 practice items. Paired tests compare the own build with the transplant.

Student	Bare	Transplant	Own build	Difference	p
Qwen3-VL-2B	13.2	29.7	31.5	+1.8	.002
Qwen3-VL-4B[a]	18.0	34.1	40.2	+6.1	10^{-40}
Qwen3-VL-8B[b]	20.3	34.7	40.3	+5.5	10^{-29}
Gemma-4-E4B	6.7	25.4	28.4	+3.0	10^{-7}
InternVL3.5-4B	9.6	27.9	30.5	+2.6	4×10^{-6}
Qwen3.5-0.8B	6.8	23.6	28.5	+5.0	7×10^{-20}
Qwen3.5-2B	9.0	21.3	32.6	+11.3	6×10^{-82}
Qwen3.5-4B	17.9	24.5	34.2	+9.7	5×10^{-60}
Qwen3.5-9B	20.5	31.7	35.8	+4.0	10^{-12}
MiniCPM-V-4.6	10.0	28.7	31.8	+3.1	10^{-8}

[a] Source student. This is a same-student restart-rule rerun, not transfer. [b] Successful second seed. The first seed reaches 23.2, below the 34.7 transplant. Students below the rule were added after the source policies were fixed. All scores use 6,000 test items.

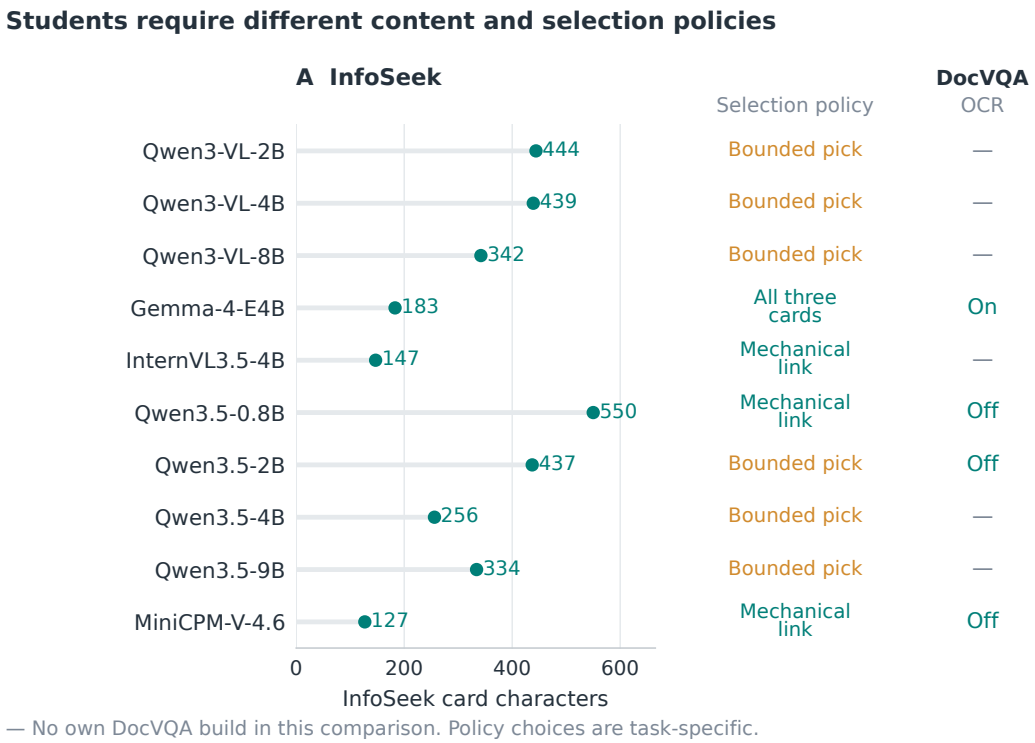


Figure 6: **Student-specific policies should be compared within the same task.** InfoSeek card widths and selection mechanisms for the ten own builds, and OCR kept/removed for the four own DocVQA builds. A dash means that no own DocVQA build is included here. The 4B policy is the restart-rule build; the 8B policy is its successful second build, whose first failure remains in Fig. 2. Unlike an average over different beds, each column describes one task.

Table 23: InfoSeek decision diagnostics for the 1,200-item source harness. Pick/net precision is measured on a 300-item probe (CLIP top-1 approximately 52%); "0" is abstention on routed test items. S uses the reference store; routed gain is measured against S' using the harness's own store.

Student	Pick / net	"0"	Routed gain	$H - S$ (p)	$S-$bare
Qwen3-VL-2B	57.0 / 55.3	1%	+3.8	+2.2 (10^{-5})	+17.1
Qwen3-VL-4B	66.0 / 58.0	17%	+7.5	+6.3 (5×10^{-36})	+15.5
Qwen3-VL-8B	61.2 / 55.7	12%	+6.6	+6.1 (5×10^{-31})	+13.8
Gemma-4-E4B	49.3 / 47.0	8%	+1.9	+3.1 (3×10^{-10})	+22.3
InternVL3.5-4B	50.8 / 48.3	21%	+0.5	+2.6 (3×10^{-6})	+19.9
Qwen3.5-0.8B	39.8 / 39.7	0%	+0.7	+3.2 (3.5×10^{-10})	+11.8
Qwen3.5-2B	87.7 / 48.7	85%	−3.9	−0.4 (0.42)	+18.4
Qwen3.5-4B	75.0 / 51.7	51%	+0.6	+2.1 (1.8×10^{-4})	+13.9
Qwen3.5-9B	76.9 / 55.3	32%	+5.8	+5.7 (1.6×10^{-29})	+12.0
MiniCPM-V-4.6	53.8 / 49.0	19%	+0.1	+0.2 (0.75)	+16.9

Table 24: Store-only (S) vs discriminate harness (H_B) on the same test items, split by whether the router fires (60%, CLIP margin small) or not (40%). "0" = share of fired items on which the student takes the escape option.

student	"0"	linked-correct on fired $S \to H_B$	acc. on fired $S \to H_B$	acc. not fired $S \to H_B$	$H_B - S$
Qwen3-VL-2B	1%	32.5→39.6	20.8→25.1 (+4.3)	44.7→43.8 (−0.9)	+2.2
Qwen3-VL-4B	17%	32.5→42.8	23.4→31.9 (+8.5)	48.7→51.7 (+3.0)	+6.3
Qwen3-VL-8B	12%	32.5→43.7	24.2→31.8 (+7.6)	49.1→52.9 (+3.8)	+6.1
Gemma-4-E4B	8%	32.5→33.0	20.0→22.4 (+2.4)	42.8→47.0 (+4.2)	+3.1
InternVL3.5-4B	21%	32.5→31.4	20.8→22.1 (+1.3)	42.7→47.2 (+4.5)	+2.6
Qwen3.5-0.8B	0%	32.5→25.1	12.9→14.3 (+1.4)	27.1→33.0 (+5.9)	+3.2
Qwen3.5-2B	85%	32.5→11.6	19.6→14.7 (−4.9)	39.2→45.5 (+6.3)	−0.4
Qwen3.5-4B	51%	32.5→26.4	22.7→23.7 (+1.0)	45.6→49.5 (+3.9)	+2.1
Qwen3.5-9B	32%	32.5→37.1	23.8→29.9 (+6.1)	45.6→50.7 (+5.1)	+5.7
MiniCPM-V-4.6	19%	32.5→33.5	18.6→18.5 (−0.1)	39.7→40.3 (+0.6)	+0.2

Table 25: Cross-task transfers on held-out test items. DocVQA uses the Gemma-compiled OCR harness; BLINK uses the 4B's fold-specific harnesses. Correspondence bare scores in parentheses use 96 rather than 16 decode tokens. Dashes denote unreported comparisons.

Student	DocVQA bare	OCR Δ	Reflectance	Correspondence	Depth
Qwen3-VL-2B	91.4	−3.7	29.9→61.9	51.3→58.1	74.2→93.5
Qwen3-VL-4B	94.9	−0.7	50.8→64.9	54.9→67.6	82.3→95.2
Qwen3-VL-8B	95.3	−0.7	–	–	–
Gemma-4-E4B	72.7	+12.2	46.3→64.2	39.5→62.6	64.5→92.7
InternVL3.5-4B	–	–	–	–	–
Qwen3.5-0.8B	84.9	−10.3	28.4→62.7	0.0 (17.9)→57.4	67.7→93.5
Qwen3.5-2B	91.5	−20.5	19.4→61.9	0.0 (19.3)→55.6	69.3→93.5
Qwen3.5-4B	94.7	−3.9	50.0→63.4	63.3→66.2	81.5→95.2
Qwen3.5-9B	95.4	−1.7	55.2→64.2	64.2→69.2	87.9→94.3
MiniCPM-V-4.6	86.1	−4.5	38.8→62.7	37.9→60.1	76.6→95.2

The correspondence rule answers the same 243/441 items at 83.1% precision for every receiving student. On deferred items, short decode budgets can prevent weaker students from returning an answer letter. A separate build for Qwen3-VL-2B returns a verdict on every item and reaches 63.3, versus 58.1 for the transplant.

Negative OCR transfer and subtractive edits. Qwen3.5-2B echoes the injected OCR block on 364/3,568 items. On clean short answers, the 0.8B and 2B lose about ten points; 73–82% of their new errors copy strings from the OCR. The teacher's own reading advantage does not make this lower-quality OCR channel helpful. In fresh builds, traced feedback removes OCR and returns performance near bare (Table 26). These builds predate the standing bare-model validation floor: the recovery was a teacher edit, not guaranteed by the original gate.

Table 26: Fresh DocVQA builds remove harmful OCR. All selected builds turn OCR off.

Student	Bare	Transplant	Initial	Selected	p vs bare	Teacher $
Qwen3.5-2B	91.5	70.9	87.0	90.8	.12	0.41
Qwen3.5-0.8B	84.9	74.5	76.3	84.6	.66	0.25
MiniCPM-V-4.6	86.1	81.6	81.0	86.4	.65	0.63

The builds use 150 probe items, 300 validation items, three reflections and GPT-5.5 without a profile. Scores are ANLS on 3,568 test items. The listed McNemar tests use ANLS thresholded at 0.5. The selected Qwen3.5-2B harness is round 3 with a parse hook. The 0.8B keeps an image-first prompt in round 1, and MiniCPM keeps a parse fix in round 2.

C.4 Evidence for decision constraints and remaining reasoning gaps

The observations below motivate limits on the tested interfaces. They do not identify an irreducible "reasoning fraction" of the teacher's advantage. If $r = \max_c \Delta_c / (\mathrm{acc}_L - \mathrm{acc}_S)$ is the fraction

recovered by the channels tested in a pilot, $1 - r$ may also reflect missing tools, imperfect retrieval, incompatible output contracts or limited search. IQ Test was screened but did not receive a comparable HC build, so it is not a measured failure of all possible harnesses. The earlier code-tool diagnostic used 2,615 MathVerse items; the WeMath comparison used 1,160 items, with advice derived from 500 practice failures. These controls motivated the taxonomy, while the main single-decision ablations provide the direct evidence for the current compiled harnesses. The practical boundary is whether useful evidence or computation can be expressed through the available interface and reliably consumed by the student; the experiments do not establish that reasoning is inherently uncompilable.

Table 27: Evidence summary for decision constraints and reasoning limits. Current results and exploratory diagnostics are separated from earlier measurements, which used different splits or runtimes and are not pooled with the main HC comparisons.

Evidence / setting	Observation	Supported interpretation
Current BLINK builds	HC trails the teacher by 14.9 points on reflectance, 20.1 on semantic and 12.3 on functional correspondence; depth is 5.6 higher (n.s.).	Available measurements support some decisions but do not close every task gap (Fig. 8).
Current compute control	CoT with voting reaches HC on reflectance and correspondence, with much higher generated-token counts (Table 7).	Some gains can also be obtained by increasing the student's reasoning compute.
Exploratory MathVista / IQ Test	MathVista: best tested tool adds 4.2 against a 19.2 teacher gap. IQ Test: teacher 51.3 versus 4B 24.0 on 150 items; no suitable carrier identified.	A teacher advantage alone does not establish a usable external channel. Unrecovered gap is not an isolated measure of reasoning.
Earlier InfoSeek lookup	Student-routed lookup: $+1.6$ versus $+5.9$ for mechanical linking; bounded candidate selection: $+6.9$.	Open argument generation and bounded selection impose different demands.
Earlier code-tool diagnostic	MathVerse: self-routed interpreter adds 10.2 for the 4B but loses 3.6 for InternVL; execution depends on the generated code.	A computation tool still requires valid, grounded arguments from its executor.
Earlier advice-block diagnostic	On WeMath, a failure-derived standing block costs the 4B 12.8 points; its benefit over generic advice is not significant.	Added guidance can interfere with reasoning under a particular model and prompt contract.

D Additional VQA results

D.1 InfoSeek: content, placement and store lineage

Store lineage: build 1 (1,200 items) rewrote all 1,380 cards in its round 0 (median card 432 characters against 496); builds 2 and 3, the DoF-menu build, the tuned human and GEPA reuse the reference store; the transplant matrix uses build 1's harness with its own store and the reference store for the store-only arm, so the matrix's $H_B - S$ includes the store difference, measured for every student by a store-only run on the rewritten store (S'): $+0.4$ (Qwen3-VL-2B), $+0.9$ (4B), $+1.1$ (8B), $+0.5$ (Gemma), $+1.5$ (InternVL), $+1.0$, -0.9, $+0.6$, $+0.2$ (Qwen3.5 0.8B/2B/4B/9B), -0.2 (MiniCPM-V-4.6); Table 23 distinguishes routed gains against S' from whole-harness gains against S. Per linking condition, accuracy when the shown card is the gold entity's rises from 36.4 (name card) to 49–52 (H_0) to 55–57 (best) and to 59.9 under the discriminate harness; on wrongly linked items it stays 9–11.

Table 28: InfoSeek $\times$ Qwen3-VL-4B: the bounded pick decomposed. S uses the reference store; S' uses build 1's rewritten store. Gains are relative to S' for build 1 and S otherwise. Parentheses report linked-correct rates before $\rightarrow$ after the pick. "Rest" means items answered without the pick.

Harness	Store	Routed	Routed gain (linking)	Rest gain	Total
Store-only S / S'	Ref. / rewritten	–	–	–	33.5 / 34.3
Tuned human pick	Ref.	100%	–	–	37.4
HC build 1 (1,200 items)	Rewritten	60%	$+7.5$ (32.5 $\rightarrow$ 42.8)	$+2.3$	39.7
HC build 2	Ref.	68%	$+8.0$ (36.7 $\rightarrow$ 46.2)	$+4.4$	40.3
HC build 3	Ref.	61%	$+4.2$ (34.0 $\rightarrow$ 44.6)	-1.1	35.6
HC, decision-menu profile	Ref.	76%	$+6.5$ (38.8 $\rightarrow$ 48.4)	$+1.8$	38.8
HC, sensitivity (3 builds)	Own / ref.	0%	No pick retained	$+0.2$ to $+0.5$	33.7–34.0

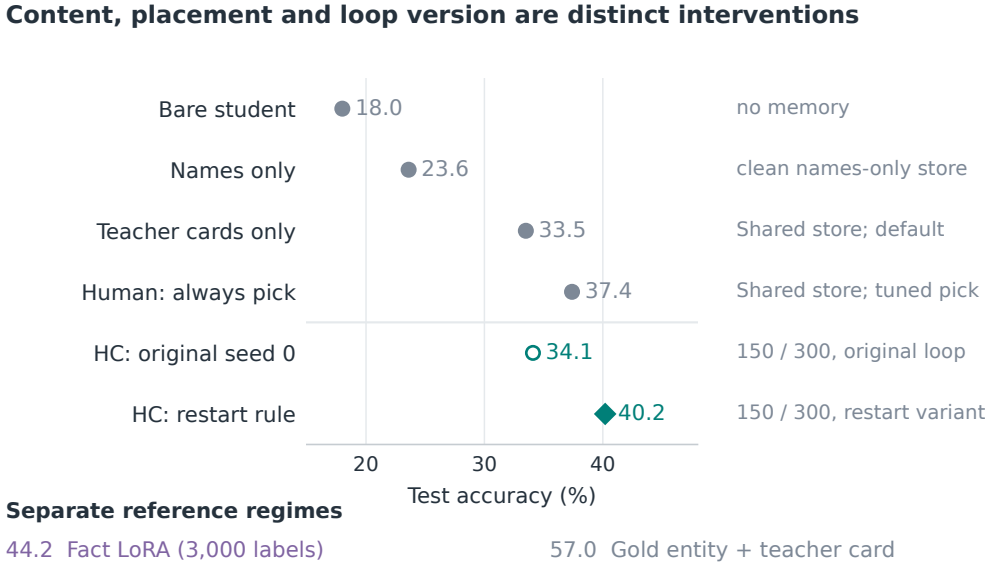


Figure 7: **Content, placement and compiler version are separate comparisons.** InfoSeek × Qwen3-VL-4B, test $n = 6{,}000$. Names-only uses the cleaned store; teacher cards and tuned human placement use the reference store. The two 150/300 builds use the same seed, splits and store, with different loop procedures; the restart-rule draw started sound without triggering a restart. Points are named arms, not an additive decomposition. The weight-trained and gold-linked references are separate regimes.

Table 29: DocVQA baselines with Gemma-4-E4B. Scores are ANLS on 3,568 test items.

Configuration	Practice items	ANLS
Bare	None	72.7
Human: plain OCR, 4,000 characters	Tuned on 500	80.1
Human: typed OCR field	Tuned on 500	82.4
Unoptimized DSPy typed program	None	86.2
One-shot teacher harness	None	85.5
HC, 50 probe / 50 validation	100	84.4
HC, 150 probe / 300 validation	450	85.6
HC, 500 probe / 500 validation	999 unique	84.9
Answer LoRA	999	84.7
Same LoRA with the 999-item HC	999	84.7
Teacher, GPT-5.5 at 2,048 pixels	Not applicable	93.3

D.2 DocVQA: consolidated baseline comparison

The earlier GPT-5.5 builds use always-on OCR, a short output contract and a parse hook; question-specific special cases are rejected by validation. Compiler-input effects are reported in Table 14, and typed-field serialization is isolated in App. E.5. Plain chain-of-thought scores 75.2; adding it to plain OCR scores 78.8 against 80.1 for plain OCR alone.

D.3 BLINK: verdict coverage and task-dependent residuals

The teacher and compiled harness are evaluated on identical items. On visual / semantic / functional correspondence, HC scores 92.4 / 54.0 / 49.2 against the teacher's 95.3 / 74.1 / 61.5; the remaining differences therefore vary substantially within the family. No compiled build significantly exceeds its task's fixed human rule.

Tool access is not enough. In the exploratory tool comparison, the 4B requests signals on 95–100% of depth, counting and spatial items, but on only 2% of visual-correspondence items. Tool-only accuracy is below the student on four of the five screened tasks. These contract-specific observations motivate the controlled activation interventions in Table 4; they are not included in the main cross-build average.

Build variation. Per-fold test scores for seed 0 / seed 1 are: reflectance 71.1, 71.1, 52.3 / 62.2, 64.4, 59.1; correspondence 72.7, 68.0, 61.8 / 64.7, 61.2, 72.2; depth 97.6, 97.6, 90.2 / 97.6, 95.1, 95.1. The selected round is round 0 in all six depth folds. On reflectance fold 2, validation prefers a round scoring 52.3 on test over the initial 56.8 (Fig. 3).

D.4 LiveVQA: retrieval, contracts and scoring

Bed. LiveVQA-2025 news sources (6,691 instances, April–May 2025), the first L1 (what is shown) and first L2 (a fact from the story) question per instance; practice 2,000 instances (probe 150, gate 300, teacher pool 150), test 1,000 instances = 2,000 questions.

Retrieval channel. CLIP ViT-L/14 image→headline linker over all articles (gold at rank 1 on 41%, in the top three on 55%); each candidate's card is its two passages most similar to the question.

Scoring and judge audit. Open answers are judged correct / incorrect / not attempted against the gold list, accuracy = correct over all items. The judge whose numbers the paper reports is Claude Sonnet 5, a model from a different provider that took no part in compiling; the same prompt run with GPT-5.5, the teacher's model (validated on 30 hand-labelled items, 28 agreements), is used as a scoring audit, alongside the LLM-free normalized containment metric used by the loop for probe and validation scores. The two judges agree on 96–99.8% of items per arm (Cohen's κ 0.85–0.98), Sonnet is the more lenient by 0.2–1.9 points on every arm, harness and non-harness alike, and the positive HC-versus-human and HC-versus-top-1 conclusions hold under all three metrics. The judge is stricter than containment on the bare and recipe arms and more lenient on the harness arms under

Table 30: BLINK verdict and fallback subsets (three held-out folds pooled). Paired scores compare the harness with bare on the same subset; a dash denotes no fallback items.

Setting	Harness	Verdict share	Verdict / bare	Fallback / bare	Total
Reflectance	HC seed 0	92%	65.9 / 50.4	54.5 / 54.5	64.9
	HC seed 1	100%	61.9 / 50.7	–	61.9
	Human	84%	60.2 / 50.4	52.4 / 52.4	59.0
Correspondence	HC seed 0	55%	83.1 / 65.0	48.5 / 42.4	67.6
	HC seed 1	65%	76.0 / 59.9	47.4 / 45.5	66.0
	Human	93%	65.0 / 55.0	53.3 / 53.3	64.2
Depth	HC seed 0	93%	96.5 / 82.6	77.8 / 77.8	95.2
	HC seed 1	94%	96.6 / 82.1	85.7 / 85.7	96.0
	Human	94%	94.9 / 82.1	85.7 / 85.7	94.4

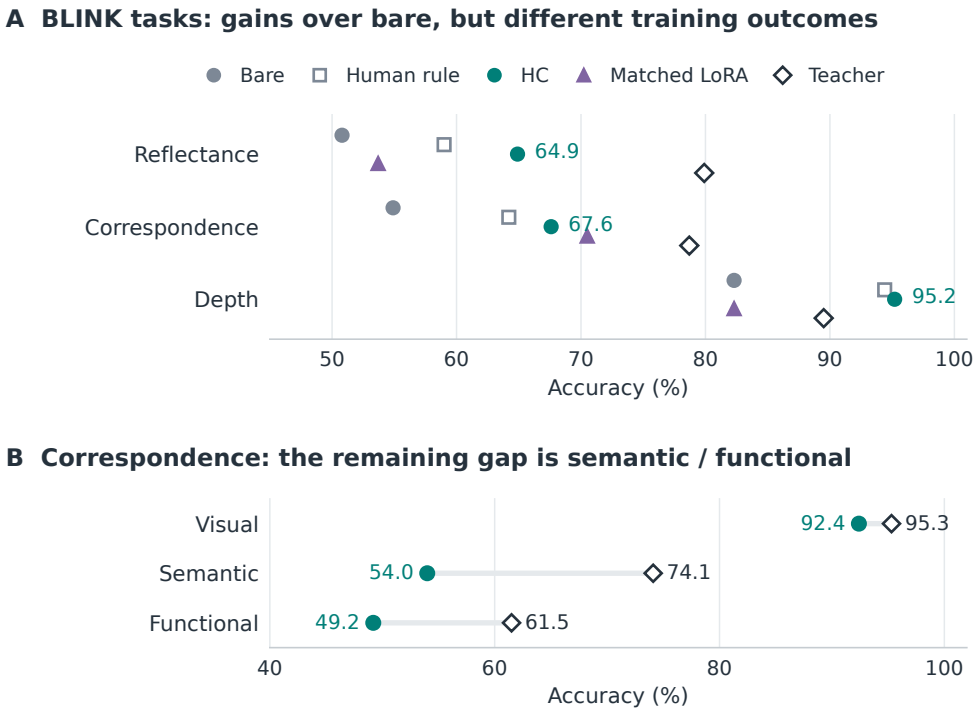


Figure 8: **Perception gains and remaining teacher gaps differ by task.** (A) Bare, human rule, HC, matched-label LoRA and GPT-5.5 at medium effort on the same items. HC and LoRA pool held-out folds; LoRA uses 89 / 291 / 82 labels per fold. (B) The correspondence subsets, not additional independent benchmarks. Semantic and functional gaps remain large; no HC build significantly exceeds its fixed human rule.

both judges, so this is a property of answer form (cards yield names in forms a string match misses), not a preference of the teacher's model for its own harness. The harness prompt has no abstention option, so the bare of Tab. 1 uses that prompt; the vqa_eval prompt with "answer Unknown" is a control (Sonnet accuracies 6.7 / 10.8 / 13.7 / 1.1 in the table's student order). GPT-5.5 with the top-three passages scores 58.3 under the Sonnet judge on a separate 300-question practice pool. **What the builds did.** Kept harnesses (no profile, three rounds, about $1 of teacher calls each): all three cards shown, a best-guess contract in place of "Unknown", a parse hook; the 9B adds a forced pick among the three headlines (its pick beats the linker), the Qwen3.5-4B shows compressed question-relevant lines of every card and never asks for a pick. The 9B's round 0 was below its bare on validation (21.3 vs 25.0) and was repaired by the first traced reflection.

Why the student-routed arm ties H_{val} here. The selected 4B harness retains reading and evidence selection with the student: it shows all three cards, rewrites each so the question-relevant sentence precedes the headline, adds a short-answer contract with a best-guess clause and a parser; select and integrate stay with the student because the reflection found no selection rule that beat the student reading all three (the retain criterion). The student-routed arm offers the same three cards with a plain instruction, so both require the student to read the supplied evidence, while differing in presentation and output contract: on the 1,081 items whose gold article is among the three they score 59.2 and 59.9, on the rest 11.1 and 9.8, 706 of 2,000 answers are identical after normalisation, and the 107 / 102 items correct in one arm only are symmetric.

Abstention changes the metric tradeoff. The one systematic difference is the contract: the compiled harness turns abstentions into guesses (60 not-attempted against 627) that are mostly wrong, so accuracy does not move but the LiveVQA F-score, which charges an abstention half a wrong answer, favours the student-routed arm (43.7 vs 37.7). The gate optimised the probe's containment score, under which a wrong guess costs no more than an abstention; a gate on the bed's own F-score would not have kept the best-guess clause. Where the compiler did remove a decision (a verdict on depth or reflectance, a page policy on SlideVQA, a bounded pick on InfoSeek) the student-routed arm gives up 5–22 points (App. C.1). Validation is 3–7 points above test for every arm including bare, so only within-test contrasts are reported. MiniCPM's numbers under the "Unknown" prompt (0.9 bare, 4.8 with the top-1 passage, 8.2 with the gold article) show a student whose failure is the abstention contract rather than the content: the same passages in the harness prompt give 20.2 containment.

Evaluation cost. Teacher edge $26 (44 answers empty at a 4k reasoning budget were re-asked at 12k), GPT-5.5 judge $25 over 22 arms with a cross-arm verdict cache, Sonnet 5 judge $16 over 35 arms, builds $4.

D.5 SLIDEVQA: PAGE SELECTION AND IMAGE BUDGET

Bed. SlideVQA (NTT release): 20-slide decks, 14,484 questions; we use the validation split (1,652) as the practice pool (probe 150, gate 300, teacher pool 150) and the test split (2,215) once. Reasoning types are derived from the annotations: numerical (has an arithmetic expression, 194), multi-hop (more than one evidence slide, 503), single-hop (1,518).

Available evidence. Tesseract OCR per slide (330 characters on average), a MiniLM question-to-OCR ranking and a CLIP ViT-L/14 question-to-slide ranking (all gold slides in the top three on

Table 31: LiveVQA test accuracy ($n = 2,000$), judged by Claude Sonnet 5. Human selects one headline; top-1 injects the mechanically linked article; gold supplies the gold article. HC beats human ($p < 10^{-27}$) and top-1 ($p < 10^{-5}$) for each student.

Student	Bare	Human	H_0	H_{val}	Top-1	Gold
Qwen3-VL-4B	14.7	25.5	28.1	37.1	27.4	51.6
Qwen3.5-4B	14.1	16.6	29.6	35.4	24.3	44.5
Qwen3.5-9B	16.9	22.5	16.1	33.9	29.9	55.1
MiniCPM-V-4.6	7.1	13.8	17.8	26.1	5.0	8.3

75–80% of single-hop and 43–46% of multi-hop questions; CLIP is two to five points above the OCR ranker).

Scoring and references. SQuAD-style exact match and token F1. Bare = the whole deck as twenty 448-px images. Human = find-then-read: the MiniLM top-3 slides at 1,024 px with their OCR and an `Answer:` field.

Table 32: SlideVQA adaptation across students. Exact match on 2,215 test items.

Student	Bare	Human	Initial HC	Selected HC	Transplant
Qwen3-VL-4B	44.7	48.7	41.9	56.7	n/a
Qwen3.5-4B	52.6	38.0	48.2	61.4	50.7
Qwen3.5-9B	51.4	47.4	56.4	62.1	63.3
MiniCPM-V-4.6	34.8	33.0	28.4	38.8	29.4
Gemma-4-E4B	43.2	35.8	38.5	48.4	43.0

Bare uses the full deck at 448 pixels. The transplant is the Qwen3-VL-4B harness without changes. Selected HC exceeds the human recipe at $p < 10^{-8}$ and bare at $p < 10^{-3}$ on every row, using paired exact McNemar tests.

Table 33: SlideVQA input controls. Gold-slide controls marked [a] use the earlier human contract.

Student	Bare, 224 pixels	Top-three images	Gold slides
Qwen3-VL-4B	32.6	48.7	58.6
Qwen3.5-4B	36.7	33.6	37.0[a]
Qwen3.5-9B	n/a	31.2	55.1[a]
MiniCPM-V-4.6	n/a	20.1	26.6[a]
Gemma-4-E4B	34.6	39.0	47.6

[a] These controls change both slide selection and the answer contract relative to the final recipe. GPT-5.5 scores 64.7 with the linker's top-three slides and 78.7 with gold slides on the 150-question practice pool. The 4B's OCR-only arm scores 27.2 and its top-one-slide arm 37.6. Evaluating the labelled arithmetic expressions gives 93.8 on the numerical subset. **What the builds did.** Round 0 sat below the student's bare on validation for four of five builds (the teachers reached for OCR and five slides, which the images-only control shows is negative for the 4B) and the traced reflections repaired every one: $H_{\text{val}} - H_0$ is $+14.7$ / $+13.1$ / $+5.7$ / $+10.4$ on test. Every kept harness fuses the MiniLM, CLIP and a lexical ranking into five slides at 1,024 px, shows OCR snippets, sets a short contract and a parse hook; none retains a student pick or a student second hop. OCR is neutral for the 4B (images-only equals the recipe), worth $+4$ to $+16$ for the three Qwen3.5/MiniCPM students, and -3.2 for Gemma-4-E4B, whose bare whole-deck reading (43.2) also beats every selection recipe; its build still adds $+5.2$ over bare by raising the gold-slide recall of the shown pages from 68% to 87% and lands at the gold-page control (47.6). The 4B's harness transfers whole to the 9B ($+11.9$, equal to the 9B's own build) and harms Qwen3.5-4B and MiniCPM, whose own builds keep shorter contracts. The numerical subset moves from 16.0 to 41.2 for the 4B with no compiled arithmetic.

Branch-support diagnostics are reported in App. G.1.

E Comparisons with alternative adaptation methods

E.1 Fixed and tuned human harnesses

Human recipes are written from the task description and available tools; thresholds are fixed in advance or tuned on development items. The same authors construct these recipes and run the compiler. BLINK rules were written before those builds finished; the InfoSeek placement grid was evaluated after the teacher builds. Fixed public-content recipes and tuned placement over teacher-written

cards are distinct comparators. **Human policies on the teacher's store and tuned grids.** InfoSeek (teacher's store, validation 600, test 6,000): a grid over margin routers $a \in \{0.30, 0.33, 0.36\} \times b \in \{0.01, 0.02, 0.04\}$, "never", "always" and a MiniLM reranker; best on validation = always discriminate (41.33) → test 37.37; the teacher's no-profile builds 42.67/41.67 → 39.73/40.27 (smaller validation→test drop). DocVQA (validation 500): OCR length $\{1000, 2000, 4000\}\times$ suffix {plain, exact-span + `Answer:` + parse}; best = 4,000 plain (81.83) → test 80.11; every exact-span variant was worse (77–79). BLINK: fixed a-priori rules (Retinex σ15 with $|\Delta| < 0.08$ → same; LightGlue projection within 8% of the diagonal else DINO argmax; Depth-Anything-V2-Small at radius 4), verdict-only: 59.0 / 64.2 / 94.4.

Table 34: Human heuristics: settings, routing and verbatim prompt text. LiveVQA scores use the Sonnet judge.

bed	setting	prompt text (verbatim)	test
InfoSeek, *recognise then read*	stage 1: the three CLIP candidate names as options, store of name cards; stage 2: the two Wikipedia passages of the picked entity most similar to the question (MiniLM), fallback to the CLIP top-1 entity on abstention; 64 decode tokens	stage 1: "Which of these is the entity shown in the image?\n{options}\n0. None of them\nAnswer with the number only."; stage 2: "Retrieved from Wikipedia (about the entity the retriever believes is shown; may be the wrong entity):" + passages + "Answer the question with a single word, number, or short phrase."	33.8
DocVQA, *typed OCR field*	page image + question + 4,000 characters of Tesseract OCR in the DSPy typed-fields serialisation; generic field extractor; 96 decode tokens	context frame "Context (OCR text of the page; may contain errors):\n{tool_text}"; objective "Answer the question about the document image with a single word, number, or short phrase, copied from the page."	82.4
BLINK reflectance	Retinex log-reflectance (σ=15) at A and B; $\|\Delta\| < 0.08$ → "same", else the lower one is darker; verdict answers the item, else the student	tool text shown to the student on deferred items: "Local reflectance estimate (brightness relative to surroundings): A = %.2f, B = %.2f"	59.0
BLINK correspondence	LightGlue projects REF into image 2; if the nearest candidate is within 8% of the diagonal, pick it; else the candidate with the highest DINOv2 3×3 cosine to REF; defer if no signal	"Keypoint matcher projects REF next to point %s" / "Feature similarity to REF: A 0.xx, B 0.xx, …"	64.2
BLINK depth	Depth-Anything-V2-Small at radius 4; larger = closer; defer if a marker is missing	"Depth estimator: relative closeness at A = %.2f, at B = %.2f (larger = closer)"	94.4
LiveVQA, *pick a headline, read it*	the three CLIP-linked article titles as options, the picked article's card (k=1)	the InfoSeek stage-1 prompt over article titles; card frame "Retrieved from your memory store (may or may not describe the right entity):\n{cards}"	25.5
SlideVQA, *find three slides, read them*	MiniLM top-3 slides at 1,024 px + 1,500 characters of their OCR; 64 decode tokens	"Answer with a single word, number, or short phrase copied from the slides. Respond in exactly this format:\nAnswer: ¡answer¿"	48.7

E.2 TEST-TIME SCALING

Table 35: Compute-only baselines (test, same items). HC references use the earlier 1,200-item InfoSeek and 999-item DocVQA builds; bare scores are in Table 1. Vote = majority over 5 sampled direct answers (9 on BLINK); CoT+vote = the same over reasoning-first rollouts; TTAug = 16 augmented views with token-level averaging (Kaya et al., 2026), paired against its own single-view control on the same backend (Δ shown). Generated-token costs for these same comparison runs are in Table 7.

bed × student	vote	CoT+vote	TTAug (Δ vs own ctrl)	H_{val}
DocVQA × Gemma-4-E4B	72.5	80.4	74.8 (+1.6)	**84.9**
SlideVQA × 4B	44.6	51.1	46.8 (+2.1, N=8)	**56.7**
InfoSeek × 4B	17.6	24.6	–	**39.7**
BLINK reflectance × 4B	50.8	**64.9**	51.5 (−1.5)	**64.9**
BLINK correspondence × 4B	54.0	**68.5**	56.9 (+2.7)	67.6
BLINK depth × 4B	82.3	85.5	86.3 (+4.0, n.s.)	**95.2**

Why these comparators. The human recipes require no teacher compilation calls; both human and compiled harnesses still incur student and tool inference costs. A third option is to spend the

student's compute at inference with no teacher and no labels. Two recent VLM methods motivate these controls. AVIS (Jeddi et al., 2026) combines visual-token pruning with self-consistency rollouts in chain-of-thought mode (temperature 0.7, top-p 0.9, majority vote), with a learned predictor choosing the number of rollouts K per query. We evaluate only fixed-K self-consistency with these sampling settings, not AVIS's learned allocation policy or visual-token pruning. Its reported policy-training procedure uses 5,000 calibration questions with 10 rollouts each, beyond our practice-data budgets. Kaya et al. (Kaya et al., 2026) decode N augmented copies of the input jointly, averaging the next-token distributions at every step (TTAug). We run TTAug at $N = 16$ where feasible ($N = 8$ for SlideVQA), using three random image transforms from the paper's high-strength list plus keyboard-typo / word-split / word-drop text augmentation with the original appended after "In other words", and the sample-and-rank variant (highest cumulative log-probability). Every SC arm is a decode option of the harness runtimes (`decode.samples`), so bare, human and H_{val} specs are sampled unchanged; every candidate rollout is stored. TTAug runs on HF transformers, so each augmented arm is paired with its own single-view control on that backend (0.4 above vLLM for Gemma, 2.5 below for the 4B at 1,024 px). Per-item TTAdapt (weight updates per test item) is out of scope. Vote normalization preserves multiple-choice answer letters.

Results (Table 35). A vote over sampled direct answers moves no bed by more than 0.9; a vote on top of any harness, human or compiled, moves nothing (DocVQA typed-fields recipe 85.6 vs 85.7, H_{val} 84.8 vs 84.9; SlideVQA 56.4 vs 56.7; InfoSeek 39.73 vs 39.73; BLINK ± 0.7). Chain-of-thought with a vote reaches H_{val} on reflectance (64.9, 29 fixed / 29 broken) and correspondence ($+0.9$, $p = 0.78$), where it wins semantic correspondence (64.8 vs 54.0) and loses visual (85.5 vs 92.4, the geometric tool's cell); it stays short on depth (-9.7, $p = 0.0075$), DocVQA (-4.5, $p = 6 \times 10^{-12}$), SlideVQA (-5.6) and InfoSeek (-15.2; $+6.6$ over bare, so reasoning does surface some entity knowledge, below the public-content human recipe), and it costs the strong 4B reader 2.2 on DocVQA. TTAug against its own control: DocVQA × Gemma $+1.6$ ($p = 2 \times 10^{-4}$; text-only $+1.5$, image-only $+0.5$ n.s., low strength $+1.8$), on the human OCR recipe $+1.1$; SlideVQA $+2.1$ at $N = 8$; correspondence $+2.7$ ($p = 0.04$ text-only); reflectance -1.5 and depth $+4.0$, both n.s. at $n \leq 134$. The matched generated-token comparison is consolidated in Table 7. TTAug additionally requires N prefills per item (5 GPU-hours per 3,568-item pass on one 3090); these prefill costs are outside the generated-token metric. Offline build costs are reported separately in App. G.3.

E.3 Retrieval and prompt optimization

Table 36: External baselines (test). GEPA uses DSPy 3.3.1, a multimodal proposer, 3,000 metric calls per run and GPT-5.5 reflection. Arrows show the unoptimised seed → the selected program.

arm	InfoSeek × 4B	DocVQA × Gemma
RAG-wiki	30.9	–
RAG-wiki (gold entity)	48.8	–
GEPA-prompt	17.9 → 19.3	82.6 → 82.9
GEPA-content (fixed store / OCR)	32.9 → 33.3	**86.2** → 86.2
GEPA-structured (two modules)	34.9 → 36.0	–

RAG. Knowledge base = the English Wikipedia article of each store entity (Wikidata sitelink → extract; 1,380 articles). RAG-wiki: CLIP top-1 → top-2 350-character passages by MiniLM question similarity, injected result-framed (30.87). RAG-wiki-lead (first 600 characters): 29.63. RAG-wiki-gold (gold entity): 48.75. RAG-web (DuckDuckGo snippets, 300-item stratified subset, 225 with results): 22.0 vs store-only 30.0 and HC 38.3 on the same items.

GEPA. DSPy 3.3.1, `dspy.GEPA` with the multimodal instruction proposer (images sent as images), train = the 150-item probe, validation = 300 disjoint practice items, 3,000 metric calls per run, GPT-5.5 reflection under a $4 cap (actual $0.32–0.95 per run, 9–20 reflection calls). Programs: prompt (image + question), content (plus the fixed store card or OCR text), and structured (Discriminate → Answer, both instructions evolved, feedback naming candidates, gold entity and the choice). Every accepted candidate sat within the 300-item gate's standard error of the seed; best-validation→test drops of 2–4 points are the winner's curse. The InfoSeek instructions memorised

probe facts ("Sigiriya … rock height is about 180 m"). Budgets: GEPA 3,000 rollouts vs HC 3,250–6,000 per build; reflection spend one third to one half of HC's compile + reflect.

E.4 GEPA OVER THE SAME EXECUTABLE ARTIFACT

The loop of §4 is one search procedure: a single lineage, three reflections on a 30-trace report, a keep-better gate. To ask whether the search matters, we replace it with GEPA's (Agrawal et al., 2026) and hold the artifact and tools fixed: the candidate is the full harness JSON (prompts, memory settings, discriminate and name prompts, `router_py`, `parse_py`), evaluated by the same runtime and student; the store is fixed (the reference teacher-written store, test 33.45 with the default harness); the reflection set is HC's 150-item probe and the validation set its 300-item gate, drawn by the same code for the same seed; the proposer is GPT-5.5 at medium reasoning reading three items with their images per call; Pareto candidate selection, strict-improvement acceptance and full-validation scoring are GEPA's own (pinned commit; `optimize_anything` with a custom proposer). Every paid call is reserved against a per-run cap of $2.5 and a global cap of $10 before it is sent; the runs stopped at $2.18–2.25 after 10–12 calls and 1,860–3,080 example evaluations. A matched HC build (seed 0, no profile, fixed store, probe 150 / gate 300) cost $1.13 in four calls and ≈1,350 evaluations.

Table 37: Search-step ablation on InfoSeek × Qwen3-VL-4B, fixed store, no profile (test $n = 6{,}000$). Parentheses in the paired row give McNemar p-values. Seed 1's HC build used a 600-item gate, so the matched difference averages seeds 0 and 2 only.

search	seed 0	seed 1	seed 2	mean (SD)
HC loop	34.1	40.3	35.6	36.6 (3.3)
GEPA	33.5	37.8	36.6	35.9 (2.2)
HC − GEPA, paired	+0.6 (.24)	+2.5	−1.0 (.037)	−0.2 (matched)

The average difference on the two matched seeds is small (HC minus GEPA: -0.2 points) and the spread between builds of either exceeds the difference between them. GEPA found the edits HC finds (a bounded discriminate step on ambiguous links, relevance-filtered cards, question-aware card lines); one seed never improved on the default on the gate, three of its code edits collapsed to 23 there, and its seed-2 router hard-codes three castle names from the probe, the probe-fitting the SlideVQA audit documents (App. G.1). These runs do not support HC search superiority: what carries the result is the executable artifact with a decision structure to edit, the store, the trace-level evidence and the gate; the search inside can be ours or GEPA's, at a factor of two in cost.

E.5 TYPED INPUT AND OUTPUT FIELDS

The DSPy comparator of Table 29 is the *un-optimised* seed program: GEPA raises its validation score from 87.3 to 90.7 and its test score by 0.0 (86.24 → 86.19); without OCR, 82.6 → 82.9. Image size is not the reason either: the seed program scores 86.25 at full resolution, and 768-px images are worth $+1.0$ to bare and $+0.5$ to the typed-OCR recipe in our runtime. The effect is the DSPy chat adapter's serialisation: a system message declaring typed input and output fields and their layout, `[[ ## field ## ]]` markers around each input, and an instruction to answer starting at the answer field and ending with a completion marker. Reproduced verbatim as a harness in our runtime it scores 81.1 without OCR (bare 72.7) and 85.3 with OCR (typed recipe 82.4), 0.9 short of the DSPy program. It is therefore an *integrate*-type decision, how the inputs and the output contract are serialised, not an optimised instruction, and we expose it as a runtime option (`prompt.format = typed_fields`) that a harness may select on any bed; App. A.2. Its effect elsewhere is student- and bed-specific (Tab. 38): $+9.0$ / $+3.3$ for Gemma, 0.0 to -0.7 for the 4B on InfoSeek, -1.3 for the 4B on DocVQA, -6.0 on reflectance; DSPy's optimiser, a search over the instruction text on 150 training items, adds $+3.4$ on validation and 0.0 on test here.

E.6 STUDENT-DRIVEN CODE AND TOOL-USE CONTROLS

We implement two agent-style controls inspired by ViperGPT and ToolLLM; these are not reproductions of their original models and training pipelines. **ViperGPT** (Surís et al., 2023) has a code model write one program per query from the question and a tool API, never seeing the image; **ToolLLM**

Table 38: Typed fields with the same items and content. Only serialization changes.

Setting	Input	Plain	Typed
DocVQA, Gemma-4-E4B	Bare	72.7	81.7
	OCR recipe	82.4	85.7
DocVQA, Qwen3-VL-4B	Bare	94.9	93.6
InfoSeek, Qwen3-VL-4B	Bare	18.0	18.0
	Name card	23.6	22.9
	Store only	33.5	33.4
	Pick recipe	25.0	24.7
LiveVQA, Qwen3-VL-4B	Bare	14.7	13.7
	Pick recipe	25.5	24.9
BLINK, Qwen3-VL-4B	Reflectance, bare	50.8	44.8
	Correspondence, bare	54.9	57.1
	Depth, bare	82.3	81.5

(Qin et al., 2024) trains a 7B model on teacher-written API-call trajectories and searches at inference with depth-first backtracking (DFSDT). What transfers to our beds is the procedure, in which the student decides which tool to call, reads the result, calls again or answers, and may backtrack: the activation, selection and argument-generation decisions studied in Sec. 3. One runtime gives the student the harness's own tools (InfoSeek: the three CLIP candidates with their store cards; DocVQA: the page OCR and a line search; BLINK: the precomputed measurements) plus a self-query, in two modes: the student writes a Python program over the API without seeing the image (*ViperGPT-S*; a failing program falls back to a self-query), or runs a ReAct loop with give-up-and-backtrack, depth ≤ 4 and ≤ 8 nodes (*DFSDT-S*, without trajectory training; a separate trained control is reported in App. F.2). Zero teacher calls. A large-teacher version using GPT-5.5 to write per-query programs is run on 150-item practice subsets under a hard cap.

Table 39: Student-driven agent controls with the same tools. Scores use the test splits.

Setting and student	ViperGPT-S		DFSDT-S	Tool use	HC
	Prose	Example			
BLINK reflectance, 4B	44.0	31.3	44.8	0%	64.9
BLINK depth, 4B	83.9[a]	78.2	81.5	0%	95.2
BLINK correspondence, 4B	42.6	51.5	58.1	0%	67.6
DocVQA, Gemma-4-E4B	75.0	n/a	75.2	81%	84.9
DocVQA, Qwen3-VL-4B	44.5	n/a	95.0	97%	n/a
InfoSeek, Qwen3-VL-4B	14.1	n/a	19.6	28%	39.7

The example column adds one example return value to the BLINK API documentation. Tool use is the share of DFSDT-S items with a non-query tool call. The three BLINK rows use Qwen3-VL-4B. Bare and human-recipe references appear in Table 1. The additional Qwen3-VL-4B DocVQA reader scores 94.9 bare. [a] Every program in this depth arm crashed on the signal structure and fell back to a self-query. The example-documentation arm has working programs.

Table 40: Teacher-written programs on covered practice items. These are not full-test comparisons.

Setting	Reasoning setting	Items	Score
DocVQA	Medium	32	81.2
DocVQA	Low	58	65.5
InfoSeek	Reported pilot	44	29.5

The DocVQA medium-effort run costs about $0.05 per program under a $1.5 cap. The low-effort run costs about $0.02 per program under a $1 cap. Each score is evaluated on the items that run covered.

Three things hold across the beds (Table 39). Student-written programs remain below the compiled harness across the evaluated tasks: on BLINK the programs read the measurements and the rules they write are inverted or anti-correlated (depth inverts the documented sign convention); on DocVQA the strong 4B reader replaces its own reading by regular expressions over noisy OCR and loses fifty points, the channel-edge rule of §5.3 inflicted by the student on itself; on InfoSeek the programs string-match the cards. The search never activates a measurement tool on BLINK (0 of 699 items; its $+3$ on correspondence is self-queries, a chain of thought) and does call OCR on DocVQA (81–97% of items) but integrates the fetched lines worse than the harness's injected text under a contract (Gemma 75.2 vs 84.9; the 4B ignores what it fetched): activation and integration are separate decisions, and the compiled harness fixes both. A large model writing the program per query reaches the store-only level on InfoSeek and, at medium reasoning, the OCR-recipe level on DocVQA on the items its programs cover (at low reasoning its DocVQA programs fall below bare), at a large-model call per query costing $0.02–0.05 each. These controls show that access to the same tools is insufficient under the tested student-driven interfaces; they do not rule out other agent training or prompting schemes.

F Data efficiency and weight adaptation

F.1 Practice budgets and the 100-item regime

At 100 practice items, HC averages 36.2 / 84.4 / 53.7 on InfoSeek / DocVQA / SlideVQA, above the corresponding answer-LoRA means. The comparison is not uniform across seeds: one SlideVQA build scores 46.6. Increasing InfoSeek trajectory-SFT training from three to ten epochs raises its 100-item mean from 30.2 to 35.3, near HC's 36.2. The original 450-item HC builds and the restart-rule build have different outcomes despite using the same number of items; budget alone is therefore not an explanation of the gap. Small validation sets also show substantial validation–test fluctuations.

F.2 Trajectory supervision at matched item budgets

For InfoSeek, GPT-5.5 writes ReAct trajectories using deterministic tools (candidate cards, lookup and finish) on the same 450 items as the original seed-0 HC build. Of 450 requests, 449 yield accepted trajectories (832 per-step rows; teacher cost $4.26). Rank-8 LoRA training takes 51 GPU-minutes. The untrained control in this trajectory-specific runtime scores 30.2; it is distinct from the DFSDT-S interface in App. E.6.

The trained agent scores 36.9, above answer-only LoRA on the same items (23.7) and the original HC builds (34.1 / 35.6), but below the later restart-rule HC draw (40.2). At 1,200 practice items, 1,199 accepted trajectories produce 40.7, statistically tied with the 1,200-item HC build at 39.7 (paired difference $+0.9$, 95% interval $[-0.1, +2.1]$, $p = .10$). The 450-item result therefore establishes an advantage over the original builds, not over every HC build at that budget. Costs and results for both HC procedures are retained.

The trained agent executes essentially a fixed program: store call then answer on 5,996/6,000 items at 450 trajectories and 5,999/6,000 at 1,200, with no substantive backtracking. These measurements support a comparison between external decision structure and decision structure learned in weights.

F.3 Answer adaptation and task-specific training protocols

DocVQA at matched labels. An answer-only LoRA on Gemma-4-E4B (rank 8, 31 GPU-minutes, no teacher) trained on the 999 practice items the Gemma build consumed scores 84.7 ANLS on the 3,568 test items against the compiled harness's 84.9 (-0.2 $[-1.3, +0.9]$, p=0.78), $+2.3$ over the typed human recipe (p=7×10^{-4}) and $+12.0$ over bare; the harness run on top of the adapter adds nothing (-0.03; its OCR text is shown on 99% of items). A thousand gold spans teach this student

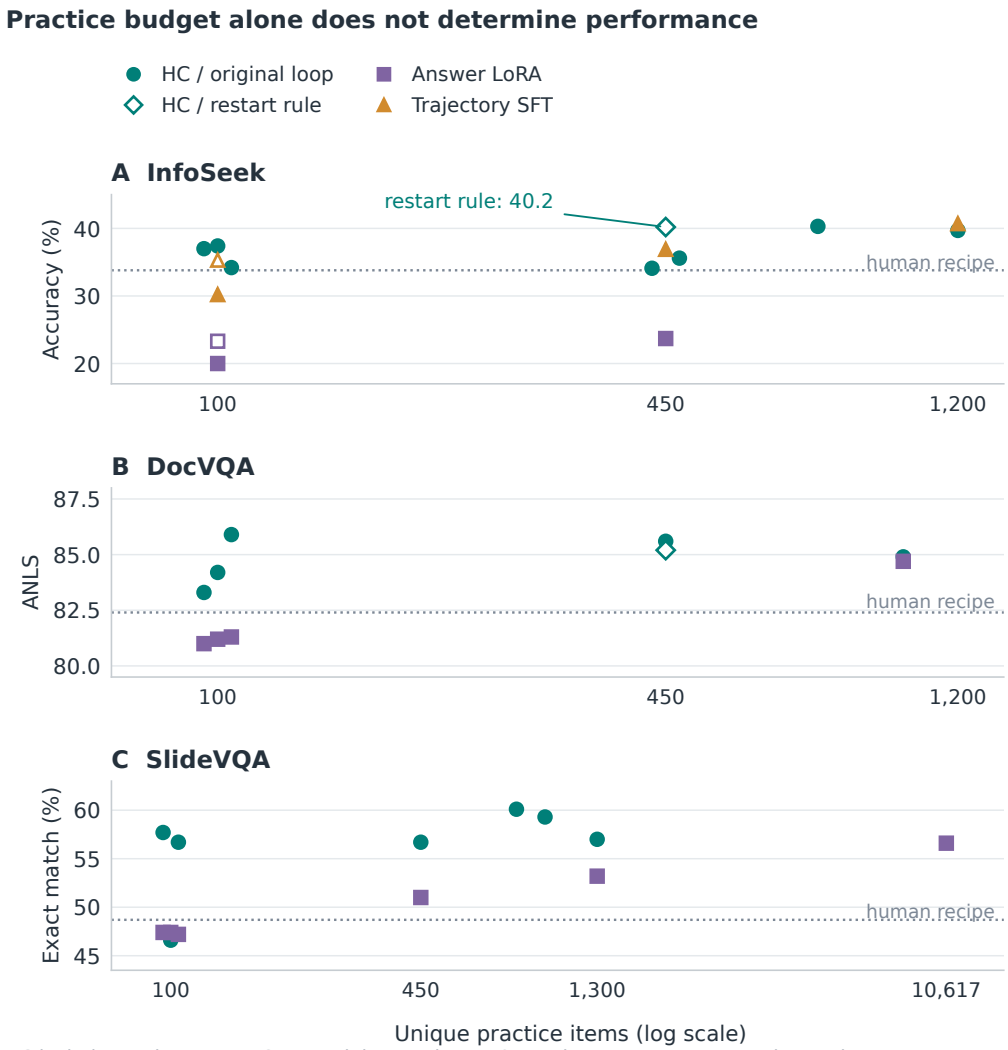


Figure 9: **Budget, build variation and loop version should not be conflated.** HC dots are individual builds; hollow diamonds use the restart-rule procedure. Training points use the same unique practice items where matched, with 100-item training means where reported; hollow training markers are 10-epoch variants. HC counts include probe and gate. Dots are slightly offset horizontally to reveal repeated seeds. Different allocations and procedures are not connected into a scaling law; full splits and accepted trajectory counts are in Tab. 41.

Table 41: Practice budgets and weight adaptation. HC counts probe / validation items. At 100 items, entries are three-seed means (individual builds: Fig. 9); epoch counts are shown when varied. Training counts are stated where they differ from the row budget. The DocVQA 500/500 setting contains 999 unique items.

Setting	HC probe / gate	HC	Answer-LoRA	Trajectory SFT
InfoSeek	50 / 50	36.2	20.0 (3 ep) / 23.3 (10 ep)	30.2 (3 ep) / 35.3 (10 ep)
	150 / 300, original	34.1 / 35.6	23.7	36.9
	150 / 300, restart rule	40.2	–	–
	150 / 600	40.3	–	–
	600 / 600	39.7	–	40.7 (1,199 accepted)
DocVQA	50 / 50	84.4	81.1	–
	150 / 300	85.6	–	–
	150 / 300, restart rule	85.2	–	–
	500 / 500	84.9	84.7	–
SlideVQA	50 / 50	53.7	47.3	–
	150 / 300	56.7	51.0	–
	500 / 300	60.1	–	–
	150 / 800	59.3	–	–
	500 / 800	57.0	53.2 (1,300 labels)	–
	Full-train reference	–	56.6 (10,617 labels)	–
BLINK reflectance	44 / 45 per fold	64.9	53.7 (89 labels)	–
BLINK correspondence	147 / 147 per fold	67.6	70.5 (291 labels)	–
BLINK depth	41 / 42 per fold	95.2	82.3 (82 labels)	–
LiveVQA (judged)	150 / 300	37.1	–	33.4 (449 accepted)

the format and the reading that the harness had supplied as a contract and a channel, and once they are in the weights the channel is redundant, as the reflectance tool was for the 20k-label LoRA.

BLINK at matched labels. For each BLINK bed an answer-LoRA (rank 8, 3 epochs, bare prompt, gold letter as target; both images on correspondence) is trained on the items of two of the harness builds' three folds (89 / 291 / 82 labels) and scored on the held-out fold, the three folds pooled and paired against bare and the compiled harness on the same items. Reflectance 53.7 ($+3.0$ over bare, $6/2$, $p = 0.29$; -11.2 below HC, $18/33$, $p = 0.049$) and depth 82.3 ($+0.0$, $8/8$; -12.9, $4/20$, $p = 0.0015$): the measured gains over bare are not significant where the harness's edge is a rule over a measurement the student cannot make. Correspondence 70.5 ($+15.6$ over bare, $88/19$, $p = 8.5 \times 10^{-12}$; $+2.9$ over HC, $53/40$, $p = 0.21$): this is also the setting where the student's chain of thought with a vote also reaches H_{val} (App. E.2) and the student-routed arm reads the raw matcher distances to 64.9: a skill the student carries, which labels or reasoning unlock and the harness supplies more cheaply, not a channel edge.

LiveVQA trajectories. The InfoSeek recipe on the news-image task: 449 accepted GPT-5.5 trajectories over the 4B build's 450 practice items ($5.8), 842 per-step rows, LoRA rank 8, DFSDT at test with deterministic tools. Judged 33.4 (GPT-5.5 judge 32.5; containment 32.1): $+18.6$ over bare, $+7.7$ over the recipe, -3.8 below H_{val} on the same practice items ($p = 5 \times 10^{-5}$; -3.2 under the GPT-5.5 judge). Every trajectory is again the fixed program (one `clip_candidates` call, then `finish`; `read_card` on 0 of 2,000 items), with one collapse the harness does not have: 126 items answered "Netflix" (6 correct).

SlideVQA at matched labels. LoRA (rank 8, LLM projections only) on the bare whole-deck input of the Qwen3-VL-4B, with the same items the harness builds consumed: 450 labels (the probe and gate) 51.0, 1,300 labels 53.2, the full 10,617-item train split 56.6, against bare 44.7 and the frozen harness 56.7 (tie at 10.6k labels, $p = 1.0$). The compiled harness run on top of the 1,300-label adapter scores 63.9 ($+10.7$ over the adapter, $+7.3$ over the frozen harness), and the full-train adapter adds nothing to that (64.1, $p = 0.86$): the combination outperforms either route alone in this setup; this does not identify exactly which components the adapter learned. MMStar moves by $+0.7$ / $+1.1$ / $+3.9$ for the three adapters, so there is no drift penalty on this bed. Label use is asymmetric: the loop reads 150 items' labels as traces and the gate labels as one number per round, whereas the adapter uses the training labels directly.

Reflectance label scaling (Fig. 10). A LoRA trained on IIW pairs in BLINK's format scores 52.2 with 44 labels, 56.0 / 50.0 with 147, 64.2 with 1,000, 72.4 with 5,000 and 80.6–81.3 with 20,000. HC scores 64.9 using 44 probe plus 45 gate items per fold. The 89-label BLINK LoRA (53.7) uses a different training source and is shown separately.

Training data: Intrinsic Images in the Wild (5,230 photos, ≈875k pairwise judgments) synthesised into BLINK's format (white ring + black tag with the letter above each point; the BLINK question verbatim; majority judgment with weight ≥ 0.5; ≤ 4 pairs per photo); 132 photos whose perceptual hash matches a BLINK reflectance image were removed (132 of 134 BLINK items covered); pool 20,001 pairs. Recipe: rank 8, α 16, lr 5×10^{-5}, batch 1 $\times$ grad-accum 16, LM projections only, vision frozen, images $\leq$768 px; 44/147/1k at 3 epochs, 5k at 2, 20k at 1 (≈6.5 s/step on one RTX 3090). The main comparison is in Section 5.5. HC on top of the 20k LoRA: transplanted harnesses 66.4; fresh 3-fold build 82.1 (teacher $2.07), the teacher removing the tool in two folds and keeping a conservative router with 91% precision on 22/45 items in the third. InfoSeek: the LoRA was trained on all 3,000 practice items, so for a fresh build a second LoRA was trained on 2,000 and probe/validation drawn from the held-out 1,000 (a distinct fresh-build protocol).

F.4 Label scaling and recompilation after tuning

Figures 10 and 11 distinguish two questions: when more training data overtakes a fixed harness, and how that harness should change after the student is tuned.

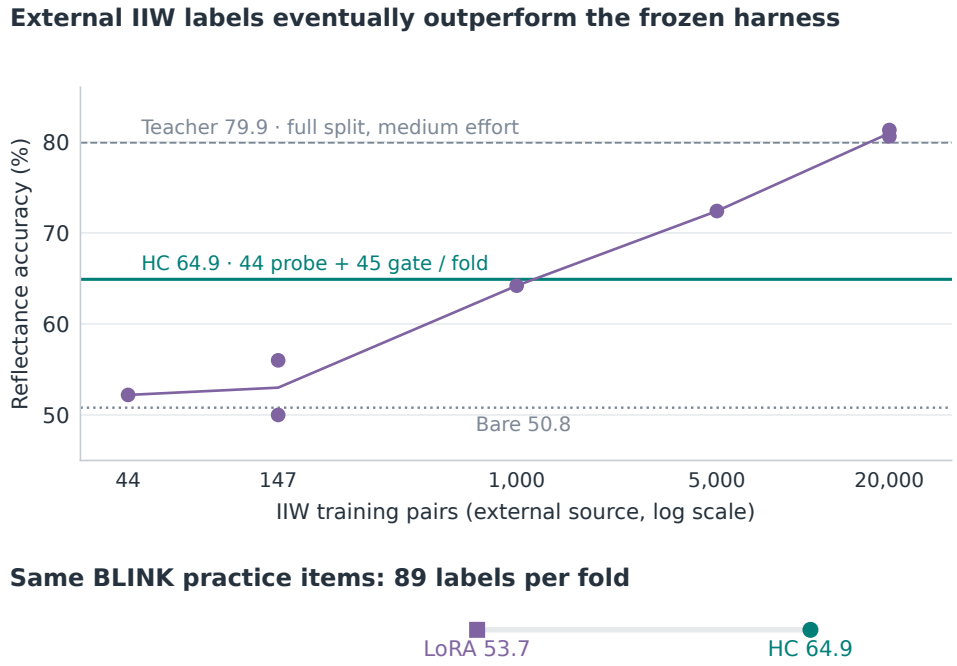


Figure 10: **Large external training sets overtake the frozen harness; matched small-fold training need not.** (A) LoRA trained on contamination-audited IIW pairs; dots show all available seeds, with a line through budget means only as a visual guide. HC uses 44 probe plus 45 gate BLINK items per fold. The teacher reference is 79.9 on the full split at medium effort. (B) LoRA trained on the same 89 BLINK practice items as HC, a distinct data regime.

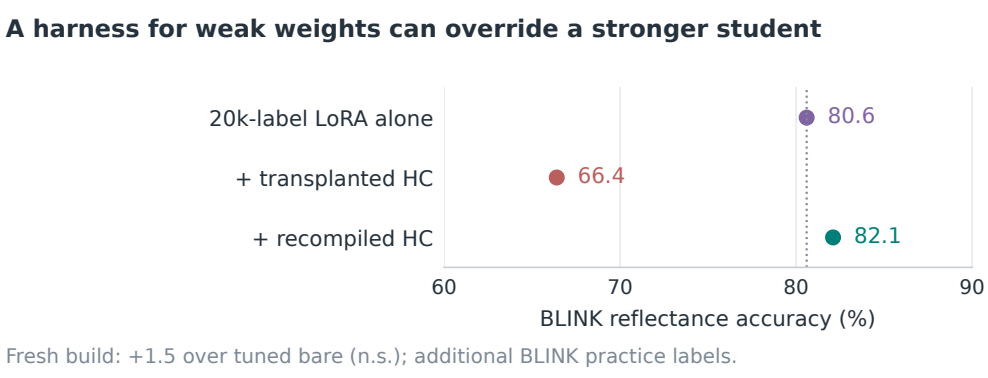


Figure 11: **Weight adaptation can reverse a harness component's value.** The old reflectance harness suppresses the 20k-label LoRA; a fresh cross-build removes the tool in two folds and uses a conservative router in the third. Recompilation consumes additional BLINK practice labels. Its $+1.5$ over the tuned bare student is not significant.

G Reliability audits and cost

G.1 Support, store and evaluation audits

For each round we count how often every router branch fires, and with what precision, on the probe, the validation set and test. The broad numeric verdicts of the 4B's round 1 fired on 11 probe items at 46% and on 19 validation items at 16%; the teacher removed them from the traces in round 2. The eight rules of its selected round 3 are keyed to phrases of specific probe decks ("exporter to Med", "credit-card handling fee"), fired on 3 probe items, 0 validation items and 0 of 2,215 test items; MiniCPM's rejected round 3 fired on 24 probe items at 96% and on one validation item. A scan of the hooks for string literals that occur in probe items finds two to four per kept harness on this bed and up to thirteen in a LiveVQA round the gate rejected; they touch zero to two test items. Two test passes settle what the rules were worth: round 3 with its verdict rules and parser special cases removed scores 56.70 against the selected 56.66 (one item differs), and round 2, before the rules were added, scores 56.98 ($p = 0.53$); the whole gain over H_0 is the ranker fusion, the contract and the parser of rounds 1–2. A subsequent runtime safeguard reports per-branch validation support to the teacher and drops branches with fewer than ten validation firings before test.

Store isolation. A legacy name-card baseline contained fact cards for 120/1,380 entities. All names-only controls cited as such were rerun with a clean names-only store; they changed by 0.3–0.6 points. Teacher fact-card generation uses entity names, not evaluation questions or answers. Store-only controls on the reference and rewritten stores separate content changes from routing changes (App. D.1).

Evaluation safeguards. Image inputs to multimodal proposers are sent as images; reasoning-mode defaults and answer-letter normalization are fixed across compared arms. LoRA comparisons requiring a fresh compiler probe use practice items held out from that adapter's training. The LiveVQA judge comparison and the distinction between containment, accuracy and abstention-sensitive scores are documented in App. D.4. Failed builds and unfavorable transfer results remain with their corresponding experiments rather than in a separate catalog.

G.2 Exploratory VLA boundary case: LIBERO-plus

LIBERO-plus (Fei et al., 2025) evaluates robustness of language-conditioned robot policies under seven perturbation types. Our exploratory setup uses SmolVLA-0.45B and X-VLA-0.9B in LeRobot 0.6.1, with disjoint 208-episode probe and validation sets and a 600-episode test: 560 perturbed episodes plus 40 added mismatched-instruction episodes. On the latter, only an abort within the first quarter of the horizon counts as a correct refusal. The runtime exposes instruction canonicalization against the suite's ten training instructions (MiniLM), OWLv2 detections, motion/stall signals and

a step budget; the teacher may write instruction overrides and stopping rules. X-VLA uses absolute end-effector control. No teacher runs online. These scores use our augmented evaluation, not the benchmark's standard leaderboard protocol.

Table 42: Exploratory policy evaluation (600 episodes). "Base H_0" is the first SmolVLA harness, reused unchanged on the other checkpoints; it is a diagnostic artifact, distinct from the validation-selected own build.

Policy	Bare	Canonical only	Base H_0	Own H_{val}
SmolVLA-0.45B	3.8	33.2	37.0	34.5
SmolVLA, tuned on perturbations	11.8	59.0	64.7	63.2
X-VLA-0.9B	59.7	60.7	63.3	–

Most of the base SmolVLA gain is recovered by removing perturbation tags from the instruction. X-VLA does not show this collapse: canonicalization adds little, and the transplanted harness adds a few points through its external control settings. Providing its decisions as appended instruction text scores 60.2. The base SmolVLA validation-selected round is worse than its first candidate on test (34.5 versus 37.0, n.s.), and object detections do not distinguish mismatched instructions reliably.

These results illustrate interface repair and a limited external-control benefit. They do not isolate long-horizon planning, recovery across subtasks, or real industrial deployment. We therefore keep this as a boundary case outside the seven-setting VQA average. A stronger robotics claim would require task-level sequencing and recovery measurements beyond aggregate perturbed-policy success. The exploratory rollout cost was approximately 19 GPU-hours; it is not included in the VQA build-cost range.

G.3 Compilation and deployment cost

Table 43: Representative VQA runtime costs. HC output tokens include all student passes; calls below one reflect direct code verdicts. CoT is a single reasoning rollout; the multi-sample CoT+vote comparison is in Table 7. The InfoSeek and DocVQA cost records correspond to the earlier 39.7- and 85.7-score builds, respectively.

Setting	HC calls/item	HC output tokens	CoT output tokens	Teacher $ / build
InfoSeek	1.6	4.9	262	6.9 (with store)
DocVQA	1.0	11.2	97	0.9
BLINK reflectance	0.08	0.1	237	2.5
BLINK correspondence	0.45	0.7	278	3.0
BLINK depth	0.07	0.1	255	1.8

InfoSeek fact-store regeneration uses 138 teacher calls (approximately 230k tokens at low reasoning effort), costing about $6; the store is reusable. Compilation and three reflections add approximately 65k tokens and $1. Thus a $1 build with an existing store and a $7 build with a fresh store are different cost regimes. These are GPT-5.5 charges for the reported configurations, not model-independent prices. Student compute is measured on RTX 3090 GPUs: roughly 25 GPU-minutes for an InfoSeek build and 1–35 for a BLINK cross-build, compared with 2.25 GPU-hours for the 20k-pair reflectance LoRA. Search, trajectory generation and judging costs are reported with their respective comparisons; deployment uses zero teacher calls. Output-token counts do not include image-prefill or tool-computation costs.

Accounting for Table 7. The CoT-with-voting comparison uses five reasoning samples on the content tasks and nine on BLINK. Generated-token counts include all sampled rollouts and all student passes. A code-only verdict contributes zero student tokens. Dividing the reported rounded means gives approximate CoT-to-HC ratios of 220, 53 and 32 on InfoSeek, DocVQA and SlideVQA. The corresponding BLINK ratios are about 20,000, 3,600 and 21,000. These are not latency ratios. Table 10 identifies the reference builds, and the single-rollout CoT costs in Table 43 are a separate comparison.