EVOVE: EVOLVING VIDEO EVALUATION THROUGH A SELF-IMPROVING HARNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern video generation models are improving at a rapid rate, and they are quickly spreading into domains such as film production, advertising, education, and robotics simulation. Each new generation brings new capabilities and, with them, new failure modes: Seedance 2.5 accepts new audio references and emits longer clips than Seedance 2.0, and its most consequential errors (*e.g.*, Audio–Visual Speech Misalignment) simply did not exist for its predecessor. Benchmarks, however, do not move with the models. Therefore, we introduce **EvoVE** (*Evolving Video Evaluation*), a model-agnostic, recursively self-improving evaluation harness for AI-generated video and multimodal understanding. EvoVE jointly evolves an evaluation plan, a trace-analysis textbook that serves as memory, and a dynamic tool bank. On error taxonomy benchmark, EvoVE achieves a macro-F1 of 88% at 5-shot, compared to the 67% VLM baseline. We formulate harness evolution as meta-learning, enabling harness self-improving. With only 5-shot target examples, EvoVE self-improves by 11.9% F1 score in-distribution and 7.8% F1 out-of-distribution. Finally, with only a task-adapter change, **EvoVE** generalizes from video evaluation to image benchmarks, achieving 82%–93% held-out accuracy on TextVQA, MMBench, and MMMU.

1 INTRODUCTION

Video-generation evaluation involves requirements specific to each generation request: a character should preserve its reference appearance, spoken words should match a script, and lip motion should follow speech. A visually coherent clip can violate any of these requirements. Evaluators must therefore accommodate changes in which defects matter and how they are identified.

Existing benchmarks provide standardized measurements of video quality and condition consistency (Huang et al., 2024; 2025; Liu et al., 2024a), but each measurement is fixed at release, and no detector exists for a newly requested category. Extending coverage is not a matter of adding a metric: it requires identifying what evidence is relevant and defining how that evidence supports a verdict. A pronunciation evaluator, for example, needs a transcript aligned with the script and rules that distinguish spoken errors from speech-recognition errors.

Agentic evaluators partially address this by orchestrating specialist measurements through a multimodal judge (Yang et al., 2025; Zhang et al., 2025), and self-improving systems go further by revising a frozen model's prompts (Agrawal et al., 2026), context (Zhang et al., 2026; Ye et al., 2026), or harness code (Lee et al., 2026), while tool-creating agents acquire new capabilities outright (Qiu et al., 2025). Each of these changes one part of the system. For video evaluation, the open question is how to coordinate all of them from labeled failures, since a missed defect may require a new measurement, a different interpretation of existing evidence, or a revised threshold.

Therefore, we introduce EvoVE, which keeps the judge frozen and jointly revises the harness around it: the evaluation plan, the error-analysis textbook, the tool bank, and the global context. A trace analyst identifies the judge's mistakes and the evidence it lacked, a tool factory acquires tools for that evidence and verifies them against held-out labels, and a planner updates the plan and context to use them. Once trained, the harness is applied to an unseen category either unchanged or after a single adaptation pass on that category's support examples, with the tool bank frozen. Building the harness thus consumes the full labeled data of the training categories, whereas adapting to a new category consumes only a handful of examples.

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.

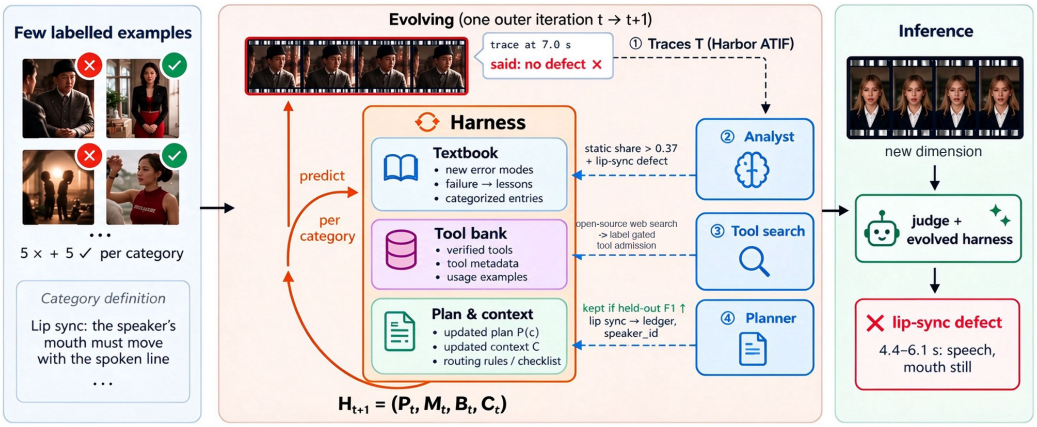


Figure 1: **Overview of EvoVE.** During construction, labeled execution traces guide updates to the error-analysis textbook, the tool bank, the evaluation plan, and the global context. At inference, the frozen judge uses the resulting harness to assess a new video against a category definition. The schematic shows the flow of information; support, development, and test access are specified in Sections 3.1 and 4.1.

We make three contributions:

- **A self-improving harness for multimodal video evaluation** in which the plan, textbook, tool bank, and context evolve together. On the training-category development comparison, the evolved harness reaches 88% accuracy against 67% for VLM baseline.
- **Meta-learning-inspired harness transfer.** Training categories supply support examples for the judge and development feedback for harness updates; the resulting harness then adapts to an unseen category from its support set alone, with the tool bank frozen.
- **Applicability beyond video-generation evaluation.** With only a task-specific adapter and development data, the same procedure reaches 82%–93% held-out accuracy on TextVQA, MMBench, and MMMU, with gains over the direct judge varying by benchmark.

2 Related Work

Video generation benchmarks and evaluation. Video benchmarks establish the dimensions and procedures along which generators are compared. VBench (Huang et al., 2024; 2025) provides dimension-specific evaluators and human-alignment data, while EvalCrafter (Liu et al., 2024a) combines objective metrics with human-aligned aggregation. T2V-CompBench (Sun et al., 2025) and VideoPhy (Bansal et al., 2025) evaluate compositional and physical plausibility, and VideoScore (He et al., 2024) learns scores from human ratings. VBench-2.0 (Zheng et al., 2025) and EntityBench (He et al., 2026) extend evaluation to intrinsic faithfulness and cross-shot entity consistency. All of these fix their dimensions and evaluators at release; EvoVE studies the complementary problem of revising the evaluation procedure for a newly specified defect category from labeled feedback.

Agentic evaluators. A second line replaces fixed evaluators with a multimodal judge that integrates heterogeneous evidence. VideoGen-Eval (Yang et al., 2025) orchestrates specialist tools through an evaluation agent, and Evaluation Agent (Zhang et al., 2025) supports promptable evaluation of visual generation. In both, the tools, routes, and decision rules are fixed for the duration of the study; EvoVE updates them from observed errors.

Self-evolving agents and harness search. A third line, from which EvoVE inherits most directly, improves an agent by editing the program around a frozen model rather than the model itself. GEPA (Agrawal et al., 2026) uses execution feedback to revise prompts, ACE (Zhang et al., 2026) evolves context playbooks, MCE (Ye et al., 2026) co-evolves skills and context artifacts,

Table 1: **Reported optimization components and evaluation settings.** Comparison with GEPA (Agrawal et al., 2026), ACE (Zhang et al., 2026), MCE (Ye et al., 2026), SkillOpt (Yang et al., 2026), Meta-Harness (Lee et al., 2026), and Alita (Qiu et al., 2025) by the harness components each method evolves and the scope of its learning. ✓ denotes demonstrated support, △ a related or partial setting, and ✗ an unreported dimension. Marks summarize the cited experimental settings, not limits on what each framework could implement.

Dimension	GEPA	ACE	MCE	SkillOpt	Meta-Harness	Alita	EvoVE
Evolved harness components							
Instructions / global context	✓	✓	✓	✓	✓	✗	✓
Memory (error textbook)	✗	✓	✓	△	✓	△	✓
Skills (evaluation plans)	✗	△	✓	✓	△	△	✓
Tool routing	✗	✗	△	✗	✓	✓	✓
Tool search / synthesis	✗	✗	✗	✗	✗	✓	✓
Label-gated tool verification	✗	✗	✗	✗	✗	✗	✓
Learning signal and scope							
Trace-level error analysis	✓	✓	✓	✓	✓	△	✓
Video + audio tasks	✗	✗	✗	✗	✗	✗	✓
Unseen categories (in-family)	✗	△	△	△	△	△	✓

SkillOpt (Yang et al., 2026) trains agent skills as optimizable parameters, and Meta-Harness (Lee et al., 2026) searches over harness code. Alita (Qiu et al., 2025) additionally acquires and synthesizes reusable tools, so tool creation by itself is not what distinguishes EvoVE. The distinction is coordination: tool acquisition is driven by the error-analysis textbook and evaluation plan, and every acquired tool is assessed against labels before it is used. Table 1 summarizes the settings reported in these works, without implying that their frameworks could not be extended to these capabilities.

3 METHOD

Inspired by the goal of rapid task adaptation in meta-learning (Finn et al., 2017), EvoVE constructs a shared evaluation harness on source categories and then adapts it to an unseen category using a small labeled support set. The judge remains frozen throughout; what is learned is the harness around it, which stores reusable evaluation plans, error-analysis memory, tools, and global instructions.

3.1 TASK AND OBJECTIVE

The goal of EvoVE is to reuse evaluation knowledge when adapting to an unseen task from a few annotated examples. Let $\mathcal{T} = \{T_1, \ldots, T_N\}$ be a set of video evaluation error categories, each treated as a task. Tasks are grouped into error families, ranging from reference-image inconsistency to lip-sync errors, and source and target tasks may belong to the same family or different ones. We split them into training tasks $\mathcal{T}_{\text{tr}}$ and unseen test tasks $\mathcal{T}_{\text{te}}$, and give each task T_i a few-shot support set S_i and a query set Q_i. In the main 5-shot setting, the support set contains five defective and five non-defective queries per category (Section 4.1).

The two phases use these sets differently. During source-harness evolution, labeled support examples are provided in the judge's context, while labeled development queries supply feedback for harness updates and checkpoint selection. During target adaptation, the analyst and planner update the harness using only the target support set, and performance is measured on held-out target queries. Let R denote the labeled development set and J the frozen judge. After T rounds of construction, the selected state is the checkpoint with the highest development F1,

$$H^* = H_{t^*}, \qquad t^* = \arg\max_{t \le T} \mathrm{F1}\big(J(\cdot; H_t); R\big). \tag{1}$$

Because R guides both the updates and the selection, its score does not estimate generalization to untouched queries; the support budgets quoted above exclude development and target-adaptation labels. When constructed from scratch, the harness starts from $H_1 = (\emptyset, \emptyset, \emptyset, C_0)$, with empty skills, memory, and tools; the warm-started main comparison is specified in Section 4.1.

Algorithm 1: EvoVE: source-harness construction and target adaptation

Input: frozen judge J; source support S and development set R; target support S_j; rounds T; admission threshold γ
Output: source harness H^*; adapted harness H'

$H_1 \leftarrow (\emptyset, \emptyset, \emptyset, C_0)$ `// empty skills, textbook, and tools`
for $t = 1, \ldots, T$ **do**
  $\mathcal{L}_t \leftarrow$ labeled traces of $J(\cdot; H_t)$ on S and R `// development feedback`
  $\mathcal{R}_t \leftarrow \mathrm{F1}(J(\cdot; H_t); R)$ `// checkpoint score`
  if $t = T$ **then break**
  $(M_{t+1}, N_t) \leftarrow \mathcal{A}(\mathcal{L}_t, H_t)$ `// memory and tool needs`
  $B_{t+1} \leftarrow B_t$
  foreach *need* $n \in N_t$, *by priority* **do**
    $\tau \leftarrow \mathcal{F}(n, R)$ `// acquire and verify a candidate`
    if $v(\tau) > \gamma$ ***or*** *the descriptive-evidence exception applies* **then**
      $B_{t+1} \leftarrow B_{t+1} \cup \{\tau\}$
  $(P_{t+1}, C_{t+1}) \leftarrow \Pi(M_{t+1}, B_{t+1}, H_t)$
$H^* \leftarrow H_{\arg\max_t \mathcal{R}_t}$
$\mathcal{L}' \leftarrow$ traces of $J(\cdot; H^*)$ on S_j, excluding each query from its own context
$H' \leftarrow$ one analyst and planner update from $\mathcal{L}'$, with B^* frozen
return H^*, H'

3.2 Harness State and Inference

A tool τ is a program that produces measurements or descriptive evidence, often using an off-the-shelf perception model such as speech recognition, OCR, or face embedding. It returns a textual summary for the judge and, where available, scalar signals for verification. The harness at round t is $H_t = (P_t, M_t, B_t, C_t)$:

- **Skills P_t.** An evaluation plan per task T_i or task family specifies tools, inspection steps, decision rules, and an optional confidence threshold. Task-level plans override family-level plans.
- **Textbook M_t.** A long-term memory of error analyses records recurring mistakes, the relevant evidence, and proposed corrections, indexed by task, family, or general applicability.
- **Tool bank B_t.** Tools built by the tool factory, stored with verification records and reading notes describing their outputs and limitations.
- **Global instructions C_t.** Instructions shared across tasks that specify evidence priorities, the handling of missing media, and confidence reporting.

For a query $x \in Q_i$ of task T_i, the judge J receives the task definition, the relevant skills and textbook entries, the global instructions, the outputs of the routed tools, and the support set S_i with x itself excluded. It returns a verdict, confidence, reason, and cited evidence; a positive verdict below the plan's confidence threshold is changed to negative. A basic verdict requires one multimodal call; runtime probes and verification checks can add calls, and Table 7 reports the measured inference cost. For pronunciation, for example, two transcripts provide the observations, the textbook records likely recognition errors, and the plan specifies which discrepancies warrant a defect verdict. Section C.2 follows one query through this process.

3.3 Harness Optimization

Each round of Algorithm 1 runs the judge, diagnoses its failures, acquires the measurements it lacked, and rewrites the plan to use them. Three operators carry out these steps in sequence.

Trace analysis. Each judgment is logged as a trajectory in Harbor's Agent Trajectory Interchange Format (ATIF) (Harbor Framework Team, 2026), recording the media delivered, tool calls and outputs, the verdict, and the label. The analyst $\mathcal{A}$ reads these trajectories, groups false positives and false negatives into error modes, attributes each to missing evidence, misread evidence, miscalibra-

tion, or misinterpretation of the prompt, and records the proposed correction in memory. Missing measurements become tool needs N_t, each specifying a target category, a candidate signal, and a priority.

Tool acquisition and verification. For each need, the factory $\mathcal{F}$ searches the web for suitable open-source tools or implements a module using available capabilities, then verifies the candidate on development samples from the source categories for which it was proposed. Each scalar signal is evaluated as a one-threshold detector: the samples are split into two interleaved folds, and a threshold and direction are fitted on one fold and scored by balanced accuracy (BA) on the other,

$$v(\tau) = \max_{s} \frac{1}{2} \sum_{i=1}^{2} \mathrm{BA}\big(s \text{ on fold } i;\ \text{threshold fitted on fold } 3-i\big). \tag{2}$$

This separates threshold fitting from scoring, although selecting the best signal and repeatedly proposing tools can still overfit the development folds. A candidate is admitted if $v(\tau) > \gamma$. A tool below the threshold may instead be retained for otherwise unavailable descriptive evidence, such as a transcript, with that status recorded in its reading note, so admission does not imply that every tool is a reliable standalone detector. Signals identified as dataset artifacts are excluded from routing. The admission study in Section E.1 reports a different, end-to-end criterion on support labels; its results do not directly validate this signal-based reward-set rule.

Planning and validation. The planner Π updates routes, decision rules, and instructions from memory and verification records. Output validation checks that every routed tool exists and carries a verification record and reading note; invalid outputs are returned for repair, and if planning still fails, the previous plan and instructions are retained alongside the valid memory and tool updates. Throughout, the current query's label is withheld from its judge input but used for feedback and scoring, while labeled support examples remain available in context.

3.4 Adaptation to Unseen Categories

The source harness H^* stores tools, skills, and textbook entries that can adapt to an unseen task $T_j \in \mathcal{T}_{\mathrm{te}}$. Without target support, H^* is applied unchanged through its family-level skills and global instructions C^*. With a support set S_j, the judge produces support traces from which the analyst and planner update the textbook and skills once, while the tool bank B^* stays fixed. Training and test tasks may share a family; cross-family transfer is evaluated in Section 4.4. These experiments test the reuse of the learned harness, not the optimization of the adaptation operators themselves. The multi-round diagnosis in Section F is reported separately from this one-update protocol.

4 Experiments

We study four questions: **Q1**, whether EvoVE constructs a stronger evaluator than existing optimizers (Section 4.2); **Q2**, which harness components and safeguards the result depends on (Section 4.3); **Q3**, whether a learned harness transfers to unseen categories and generators (Sections 4.4 and 4.5); and **Q4**, whether the same loop applies to other multimodal tasks (Section 4.6).

4.1 Experimental Setup and Scope

Benchmarks. The benchmark inventory contains 1,886 generated videos from five generators (Section A), annotated under a taxonomy of 27 error categories in seven families: reference consistency, speech and audio, text, instruction following, spatial, visual quality, and video editing. The main comparison uses five source categories, the transfer study reports six source–target settings rather than an exhaustive sweep, cross-generator experiments cover four generators, and the public-benchmark extensions use TextVQA, MMBench, MMMU, and VBench++ human preference.

Support budgets and metrics. In the main comparison, 0-shot, 5-shot, and 10-shot denote in-context support of zero, five defective plus five non-defective, and ten defective plus five non-defective examples per category. These counts exclude development labels and labels used to construct a warm-start harness, so 0-shot does not mean label-free construction. On unseen categories, 0-shot

Method	0-shot	5-shot	10-shot
VLM (frozen judge)	65.8	67.6	67.9
GEPA	74.3	82.9	81.7
ACE	72.4	71.8	72.4
MCE	70.2	77.4	–
SkillOpt	78.0	73.5	79.2
Meta-Harness	74.0	–	–
Alita	79.2	71.1	76.4
EvoVE	**85.8**	**88.6**	**92.8**

Table 2: **Evaluator construction: development comparison.** Macro F1 percentages on development evaluation rows across five source categories at the listed support budgets. MCE has no reported 10-shot variant; Meta-Harness has no separately reported few-shot variants.

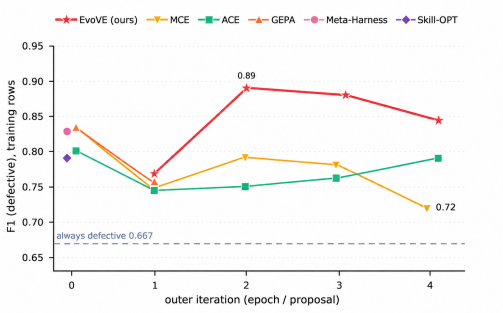


Figure 2: **Optimization trajectories.** Defective-class F1 score on source rows used by each method's loop. Update units and compute differ across methods, so the horizontal axis does not compare optimization cost. These curves are not held-out performance.

denotes no target examples and 5-shot denotes adaptation on five examples per class. We report defective-class F1, macro-averaged across categories unless stated otherwise; failed judge calls count as wrong, and intervals, where reported, follow Section B in the Appendix.

Evaluation access. The intended protocol groups shared requests, references, and near-duplicate prompts within a split. In practice, source-category evaluation rows guided several methods during development, with exposure not quantified separately by method. We therefore report Tables 2, 3 and 10 as development comparisons rather than estimates of generalization to untouched queries, and reserve the unseen-category and cross-generator experiments as held-out evaluations.

Baselines. Within a comparison, every method uses the same frozen VLM judge, query media, definitions, and support configuration. The baselines optimize prompts (GEPA (Agrawal et al., 2026)), context (ACE (Zhang et al., 2026)), skills (MCE (Ye et al., 2026), SkillOpt (Yang et al., 2026)), harness code (Meta-Harness (Lee et al., 2026)), or tools (Alita (Qiu et al., 2025)); a direct judge provides a reference. Baselines use their original implementations, so tools, initialization, and optimization budgets differ: most are scored after one epoch, round, or iteration, MCE after four iterations, and EvoVE after five (see Section B). The comparison is therefore of complete systems and does not isolate the optimization mechanism.

Implementation. The analyst, tool factory, and planner use Claude Opus 5.5 through the Claude Agent SDK. Construction uses five iterations, admission threshold $\gamma = 0.55$, and at most four tool needs per iteration. The main comparison uses Gemini-3-Pro as judge and a warm-started EvoVE harness; training trajectories and unseen-category studies use Seed-2.1-Lite and start from the seed harness. Judge-backbone experiments and other extensions specify their judges separately, and we compare numbers only within a common judge configuration. Table 7 reports measured costs.

4.2 Q1: Evaluator Construction

Reported performance. EvoVE obtains the highest macro F1 at every support budget (Table 2): 85.8%, 88.6%, and 92.8% at 0-, 5-, and 10-shot, against 82.9% for the strongest baseline (GEPA, 5-shot) and 65.8–67.9% for the direct judge, which barely moves with more examples. The gains concentrate where a measurement is needed: EvoVE leads on pronunciation (96.6%), lip synchronization (96.3%), and text rendering (96.0%), but not on the two reference-consistency categories (Table 10). Unequal budgets and prior data access (Section 4.1) limit how much of this difference can be attributed to the optimization mechanism.

Optimization trajectories. On its own optimization rows, EvoVE rises from approximately 77% at iteration 1 to 89% at iteration 2, then declines to 84% by iteration 4 (Figure 2), which is what

Table 3: **Component removal from the final harness.** Reported results (%) on evaluation queries, with five support examples per class and no retraining. Δ is the difference from the full harness in percentage points. These are development comparisons (Section 4.1); paired uncertainty estimates for the differences are not reported.

Configuration	What changes	Acc.	Δ	F1	Δ
No learning					
Seed harness	iteration 1, before any update	69.6	-18.8	76.4	-13.0
Remove one learned component					
− Dynamic tools	the 6 factory-built tools removed	83.9	-4.5	85.5	-3.9
− Memory	error-mode textbook removed	84.8	-3.6	86.2	-3.2
− Planner	seed plan and rubric served instead	84.8	-3.6	86.4	-3.0
− Recursion	harness after one update (iteration 2)	87.5	-0.9	88.3	-1.1
− Trace checks	clause and receipt checks switched off	89.6	$+1.2$	88.6	-0.8
Relax a safeguard					
+ Ungated tool	tool rejected on the labels, admitted anyway	90.2	$+1.8$	90.6	$+1.2$
Full EvoVE		**88.4**	–	**89.4**	–

motivates checkpoint selection (Equation (1)). Because these curves use optimization rows, unequal update units, and a different judge from Table 2, they do not establish generalization.

4.3 Q2: Component and Tool-Admission Analysis

We remove components from the final harness and re-score each variant with the same judge, support examples, and evaluation queries, without retraining (Table 3). This measures how much the final harness depends on each component at inference; it does not decompose their contributions during learning, and the exposure limitations of Section 4.1 apply.

Dependence on learned components. We see that learning, not tool availability, accounts for the gain: the full harness reaches 88.4% accuracy, while the seed harness scores 69.6% whether or not it is given all 17 library tools. Removing the factory-built tools, the textbook, or the learned plan lowers accuracy by 4.5, 3.6, and 3.6 points respectively, so the harness depends on all three, although these drops are not additive contribution estimates. Most of the benefit arrives in the first update: the one-update checkpoint is within 0.9 points of the final harness. Two safeguards trade off differently: removing trace checks raises accuracy but lowers F1, and admitting a tool the gate rejected raises both. Without paired intervals and repeated runs, these differences do not establish a reliable ranking of components or safeguards.

Tool-admission behavior. The separate admission study in Table 13 shows that the gate's effect depends on the judge: gated versus ungated F1 is 75.9% versus 76.3% for Gemini, 76.0% versus 75.0% for Seed, and 64.9% versus 71.4% for Qwen. The gate rejects useful as well as harmful tools, so these results do not show a uniform advantage. Note that this study uses an end-to-end criterion distinct from the signal-based construction rule (Section 3.3 and Section E.1 in the Appendix).

4.4 Q3: Self-Improvement on Unseen Error Categories

We construct EvoVE on one to three source categories, starting with no tools, memory, or skills, and apply it to an unseen category either unchanged (0-shot) or after adaptation on five defective and five non-defective examples with its tool bank frozen (5-shot). In-distribution settings stay within one error family; out-of-distribution settings cross families.

Zero-shot generalization. A harness learned on one category transfers to unseen categories in the same family without any target labels. A harness trained on pronunciation reaches a zero-shot F1 of 82.8% on lip-sync evaluation, and one trained on face-reference inconsistency reaches 83.3% on the multi-character category. Inspection of the learned harnesses suggests why: related

Table 4: **Few-shot adaptation on unseen error categories.** After zero-shot transfer, five defective and five non-defective target examples are used for adaptation. F-1 is shown in percentage points.

Setting	Source categories	Unseen target	0-shot	5-shot	Δ F1
In-dist.	Pronunciation	Lip sync	82.8	**85.7**	**+2.9**
	Speech (3 subclasses)	Lip sync	83.3	**95.7**	**+12.4**
	Face reference	Multi-character	83.3	**95.7**	**+12.4**
	Anatomy	Physics	58.7	**78.6**	**+19.9**
				Average self-improvement	**+11.9**
Out-of-dist.	Speech (2 subclasses)	Text rendering	**83.7**	83.3	-0.4
	Reference (2 subclasses)	Audio reference	47.6	**63.6**	**+16.0**
				Average self-improvement	**+7.8**

categories reuse tools such as ASR, while the textbook stores modality-specific evaluation knowledge. These observations are consistent with reuse of learned components, although they do not isolate its contribution.

Few-shot harness adaptation. With only 5-shot target examples, the evolved harness rapidly adapts to new failure modes. In-distribution transfer benefits most from adaptation, with an average improvement F1 of 11.9%. For out-of-distribution transfer, the gain is smaller, averaging F1 of 7.8%, suggesting that adaptation becomes more challenging as the source–target gap increases. Some categories, such as text rendering, show little or no improvement. Overall, these results demonstrate that with only a small number of annotated examples, the harness can recursively self-evolve to adapt to new tasks and error categories.

4.5 Transfer Across Video Generators

We construct EvoVE separately on each of four generators from the same seed harness, then freeze each evolved harness and evaluate it on every generator with five support examples per class from the target generator in context (Table 11). Diagonal balanced accuracies are 74.3% for Happy Horse, 72.5% for Seedance, 92.8% for Veo, and 80.3% for Wan. A harness can transfer across generators, and sometimes outperforms the native one: the Happy Horse harness achieves the highest score on Seedance (85.0%). Transfer varies substantially by source, however, and with only 14–54 test videos per target generator and single runs, this study cannot separate generator-specific rules from data composition and sampling uncertainty.

4.6 Q4: Applicability Beyond Video Defect Detection

We apply EvoVE to TextVQA (Singh et al., 2019), MMBench (Liu et al., 2024b), MMMU (Yue et al., 2024), and VBench++ human preference (Huang et al., 2025). For each benchmark, ours evolves a task-specific harness using only an adapter, a scoring function, and development labels; test queries remain unseen during evolution.

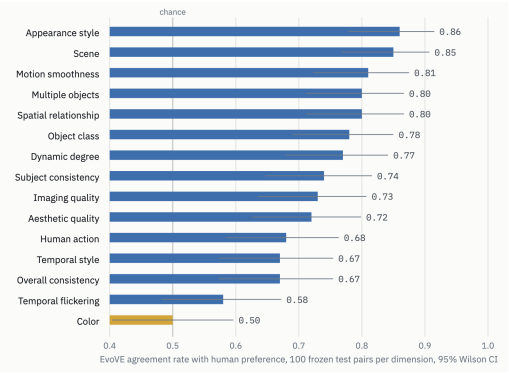
VLM benchmarks. The same loop, unchanged apart from the adapter, reaches 82.1–93.3% accuracy on 300 frozen held-out examples per benchmark across TextVQA, MMBench, and MMMU. Held-out gains over the direct judge are modest and vary by benchmark, including a small decrease in one TextVQA setting, so the result establishes general applicability rather than a consistent improvement on tasks where the judge is already strong.

Human preference. We further evaluate EvoVE on human-annotated pairs from VBench++ (Huang et al., 2025). Given two videos generated from the same prompt, the evaluator must predict which one human raters preferred on a given dimension. We cover 15 dimensions with 100 frozen test pairs each (1,500 in total). For each dimension, the harness evolves from only 20 labeled pairs: 4 in-context examples, 8 fit pairs whose judge traces drive the updates, and 8 validation pairs for selecting the final iteration.The evolved harness reaches an average agreement of 73% with human preferences(fig. 3), ranging from 86% on Appearance style to 50% on Color. Agreement is highest for Appearance style

Table 5: **Agreement (%) with each benchmark's gold answers.** *Rows*: labeled rows available to the loop (base, then pool budget). *All*: every annotated row; *Test*: 300 frozen held-out rows. Δ: paired gain over the same judge without a harness.

		All rows		Test (300)	
Benchmark	Rows	EvoVE	Δ	EvoVE	Δ
TextVQA	125	**85.3**	**+3.6**	84.1	+3.0
	1,005	82.8	+1.5	82.1	−0.5
MMBench	500	**95.3**	**+1.5**	93.0	+0.3
	1,100	**96.1**	**+3.2**	93.3	+1.0
MMMU	270	88.8	+1.2	89.7	+0.7
	600	83.6	+1.4	82.7	+0.0

Figure 3: **VBench++ agreement with human preference.**



(86%), and Scene (85%) and Motion smoothness (81%) and lowest for Temporal flickering (58%) and Color (50%). The Color dimension is at chance because 68% of its human labels are ties.

5 Discussion

5.1 Trace-Based Diagnosis

Because every judgment is exported in Harbor's Agent Trajectory Interchange Format (ATIF) (Harbor Framework Team, 2026), the harness can be diagnosed offline: deterministic code extracts inspection and evidence signals, an analyst proposes explanations and a verifier checks numerical claims. This workflow is separate from Algorithm 1. Section F reports an extended adaptation episode and exploratory runtime checks; these do not establish the configuration used for every main result.

5.2 Limitations

Each configuration is a single run, with 12–30 queries per category in the main comparison and 14–54 videos per cross-generator test column. Evaluation exposure, unequal budgets, and warm-starting limit the development comparisons; unseen-category tests lack matched target-task controls; and tool admission has mixed effects across judges. The full runtime also averages 2.1 judge calls per query against one for the direct judge. These limitations motivate untouched evaluation data, matched resources, and repeated runs.

6 Conclusion

We introduced EvoVE, a recursively self-improving harness for multimodal video evaluation in which the plan, textbook, tool bank, and instructions evolve together around a frozen judge. Jointly evolving these components substantially outperforms routing the same tools through a fixed seed harness, with the largest gains on speech and text errors that the judge cannot assess from pixels alone. Casting harness evolution as meta-learning lets the learned harness adapt to unseen error categories from a few annotated examples, improving five of six transfer settings, and changing only the task adapter carries the same loop to general VLM benchmarks. Untouched test data, matched optimization budgets, and repeated runs remain necessary to determine when joint evolution provides a reliable advantage.

Reproducibility Statement

The optimization loop is specified in Algorithm 1 and the experimental configurations in Sections B and 4.1. Reproduction requires the split manifests, exact model identifiers, baseline revisions, operator instructions, tool implementations, and selected harness checkpoints. The planned release includes per-query traces, memory entries, tool verification records, plan revisions, operator transcripts, and all code along with the video dataset.

AI USE STATEMENT

Claude assisted with the experiment design and parts of the code. Within the actual loop, Claude agents act as the analyst, tool factory, and planner; the judge model varies by experiment, as specified in the setup and table captions. Codex assisted with manuscript editing and grammar. The authors remain fully responsible for all experimental claims and protocol descriptions.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. Videophy: Evaluating physical commonsense for video generation. In *International Conference on Learning Representations*, volume 2025, pp. 102075–102121, 2025.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, 2026. URL `https://doi.org/10.5281/zenodo.20953922`.

Ruozhen He, Meng Wei, Ziyan Yang, and Vicente Ordonez. Entitybench: Towards entity-consistent long-range multi-shot video generation. *arXiv preprint arXiv:2605.15199*, 2026.

Xuan He, Dongfu Jiang, Ge Zhang, Max Ku, Achint Soni, Sherman Siu, Haonan Chen, Abhranil Chandra, Ziyan Jiang, Aaran Arulraj, et al. Videoscore: Building automatic metrics to simulate fine-grained human feedback for video generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2105–2123, 2024.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, et al. Vbench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

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.

Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. Evalcrafter: Benchmarking and evaluating large video generation models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22139–22149. IEEE, 2024a.

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

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8309–8318. IEEE, 2019.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8406–8416. IEEE, 2025.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Yuhang Yang, Ke Fan, Shangkun Sun, Hongxiang Li, Ailing Zeng, FeiLin Han, Wei Zhai, Wei Liu, Yang Cao, and Zheng-Jun Zha. Videogen-eval: Agent-based system for video generation evaluation. *arXiv preprint arXiv:2503.23452*, 2025.

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution. *arXiv preprint arXiv:2601.21557*, 2026.

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

Fan Zhang, Shulin Tian, Ziqi Huang, Yu Qiao, and Ziwei Liu. Evaluation agent: Efficient and promptable evaluation framework for visual generative models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7561–7582, 2025.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, et al. Vbench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

A BENCHMARK

This section describes the multi-generator video benchmark (Table 6).

Table 6: **Synthetic multi-generator benchmark: rendered videos per genre and generator** (1,886 in total). Requests come from two campaigns (10–30 s clips, then 30 s clips); Seedance 2.5 renders every request, and the other four generators render subsets.

Generator	Short film	TikTok	Advertisement	Gaming	Education	**Total**
Seedance 2.5	412	124	124	82	82	824
Happy Horse 1.0 R2V	87	56	69	64	53	329
Wan 2.7 R2V	104	51	66	59	49	329
Veo 3.1	150	43	51	35	35	314
Vidu Q3 Pro	38	12	18	10	12	90
Total	791	286	328	250	231	1,886

Each request comes with its own reference images, audio, and video, generated from the request itself. No real footage or real person is involved. Annotators give every video a binary label and a short explanation for each error dimension.

B EXPERIMENTAL SETUP

This section expands the setup of Section 4.1.

Splits. Each category has a support set of k defective and k non-defective examples, a development set, and a test set. Requests that share a reference image or a near-duplicate prompt stay in one split. No such group crosses a split or the training/unseen boundary. On an unseen category, the evolved harness is first scored frozen. In few-shot adaptation, the analyst and planner read leave-one-out traces on the support set and update memory and routing; the tool bank stays frozen.

Metrics. We report macro F1, the defective-class F1 averaged over categories, together with balanced accuracy, precision, and recall. Failed judge calls count as wrong. Intervals are 95% paired bootstrap intervals over queries (3,000 resamples). Every result is a single run.

Comparison scope. All methods share the frozen judge, the media, the category definitions, and the support examples. Each baseline runs its original code and evolves only its own logic: prompts (GEPA), a context playbook (ACE), skills (MCE, SkillOpt), harness code (Meta-Harness), or tools (Alita). GEPA, ACE, SkillOpt, Meta-Harness, and Alita are scored after their first epoch, round, or iteration; MCE after four iterations; EvoVE after five. Tool outputs and compute are therefore not matched across methods.

Implementation. The analyst, tool factory, and planner are Claude agents (Claude Opus 5.5) run through the Claude Agent SDK. EvoVE uses admission threshold $\gamma = 0.55$ and at most four tool needs per iteration. The judge is Gemini-3-Pro for the main results and Seed-2.1-Lite for the training curves and transfer studies.

B.1 EVALUATION AND OPTIMISATION COST

Table 7 reports what each method spends. Judge calls are counted at the shared judge proxy, and operator turns count analyst, tool-factory, and planner calls. Wall time is measured with at most four judge calls in flight, the rate the upstream quota sustains. It is comparable across methods but is not a lower bound.

C HARNESS DETAILS

This section lists the tools of the reported harness and follows one test query through it.

Table 7: **Evaluation and optimisation cost** on the selected quality benchmark at 5-shot . Inference: judge calls and seconds per test query, probes and tool calls included. Evolution: optimiser turns and wall time to the checkpoint reported in (GEPA after its first four proposals, ACE and SkillOpt after epoch 1, Alita after round 1, MCE after four iterations, Meta-Harness after iteration 1). MCE and Meta-Harness log runs or messages rather than turns; Meta-Harness's inner judge calls are not logged and its 112 test queries all ended in execution errors.

	Inference (per query)		**Evolution**	
Method	Judge calls	Seconds	Operator turns	Wall (h)
Direct VLM	1.0	83	0	0
VLM few-shot	1.0	125	0	0
GEPA	1.0	302	5	6.3
ACE	1.0	197	22	7.6
MCE	1.0	–	4 runs	8.3
SkillOpt	1.0	127	14	5.4
Meta-Harness	$\geq$1.0	83	474 msgs	4.1
Alita	1.0	306	58	3.6
EvoVE	2.1	274	4	8.3

C.1 Tool Bank

Table 8 lists the 23 tools of the reported harness. Seventeen come from the shared library; the loop routed and verified them. Six were built by the tool factory during evolution. A tool is admitted for a category only after label-gated verification on that category's support rows. Admission tests a tool as the judge uses it, alongside the video, not as a standalone classifier; tools with modest standalone scores can therefore still add evidence.

C.2 A Worked Example: One Pronunciation Query

We follow one held-out test query of the pronunciation category through the reported harness (plan version 14). The query was never shown to any operator. Every value below is read from the logged prediction record.

What each component contributes. The plan decides what to compare: spoken words against the script, not mouths or timbre. The tools supply evidence the judge cannot produce reliably by listening once: two independent transcripts with timestamps. The memory and the plan's decision rule carry what earlier iterations learned: iterations 1 and 2 added the swallowed-syllable check that fires here. The judge makes the final call and must account for every clause of the definition.

Why the judge stays in the loop. The three pronunciation gates target substituted words, the most common error on the support rows, and report nothing for a missing syllable. The judge, guided by the plan, still catches it. This is why EvoVE routes tool evidence to the judge rather than letting a gate decide alone.

D Additional Results

This section gives per-category results and three transfer studies that support Sections 4.2, 4.4 and 4.6.

D.1 Per-Category Results

Table 10 breaks the main 5-shot result down by category. EvoVE is best on pronunciation, lip sync, and text rendering, where a measurement such as speech recognition or OCR adds evidence the judge cannot read from pixels.

Table 8: **Tools EvoVE harness propose** (best iteration, 23 tools). *Admitted for*: the training categories on whose support rows the tool passed label-gated admission. *BA*: 2-fold balanced accuracy of the tool's own signal on those rows, as recorded in the tool bank. Six tools score below 0.60 on their own signal.

Tool	What it measures	Admitted for	BA
Library tools, routed and verified by the loop (17)			
`text_density_gate`	Counts distinct on-screen text regions and subtitle frames; flags text-heavy videos.	Text rendering	1.000
`text_instruction_gate`	Glyph gate (text-heavy vs. sparse) plus a check of requested or forbidden captions against the caption layer.	Text rendering	1.000
`ocr_rapid`	RapidOCR inventory of on-screen strings with confidence; flags garbled text.	Text rendering	0.938
`mouth_line_ledger`	For each scripted line, whether a visible mouth moves while it is spoken; aggregates to convict or acquit.	Lip sync	0.927
`pron_name_subclass`	Classifies script-vs.-transcript differences (names, rare words) into convictable mispronunciation types.	Pronunciation	0.899
`lip_sync`	Mouth-opening statistics while a face is visible during speech (share closed or static).	Lip sync	0.763
`speaker_identity`	ArcFace identity tracks over the whole clip, bound to reference images and scripted speakers.	Face ref., Char. appearance	0.750
`delivery_mode_gate`	Detects lines scripted as voice-over, inner monologue or closed-mouth delivery, which excuse no lip motion.	Lip sync	0.735
`ref_person_match`	Binds each reference image to a character track and reports face similarity per binding.	Face ref.	0.720
`text_align`	Aligns the scripted dialogue with a Whisper transcript; mismatched words are mispronunciation candidates.	Pronunciation	0.692
`ref_delivery_gate`	Audits which reference images were actually delivered and whether they contain a face.	Face ref., Char. appearance	0.663
`clip_drift`	CLIP similarity of reference images to frames and between consecutive frames (appearance drift).	Char. appearance	0.646
`face_insight`	InsightFace similarity between each reference face and the best-matching face in the video.	Face ref.	0.623
`scene_attribute_audit`	Checks the prompt's scene constraints shot by shot (descriptive ledger).	Face ref., Char. appearance	0.615
`asr_funasr`	Second, Mandarin-specialised transcript with timestamps, used to corroborate mismatches.	Pronunciation	0.594
`ref_appearance`	Person-crop CLIP similarity and detail ratio against the reference images.	Char. appearance	0.594
`ref_role_scoping`	Separates principal-character references from scene and prop references.	Face ref., Char. appearance	0.592
Built by the tool factory during evolution (6)			
`ocr_text_stability`	Counts screen regions whose OCR reading changes across frames (unstably rendered glyphs).	Text rendering	0.833
`cjk_lexicon_plausibility`	Counts high-confidence OCR strings that are not plausible Chinese words.	Text rendering	0.791
`pron_claim_gate_v2`	Filters mispronunciation claims, exempting ASR artefacts and homophones.	Pronunciation	0.603
`min_receipt_gate`	Lists the zoomed views a verdict must cite before it is accepted.	Face ref., Char. appearance, Lip sync	0.585
`dual_asr_consensus`	Word mismatches on which two ASR engines agree.	Pronunciation	0.522
`probe_ledger_check`	Ledger of the media the judge actually fetched, with send-back rules for uncovered verdicts.	Face ref., Char. appearance, Lip sync, Text rendering	0.500

D.2 Cross-Generator Transfer

We ask whether the harness learns a procedure that is independent of the generator. For each of four generators, EvoVE is evolved from the same untrained seed on that generator's videos only. The frozen harness is then scored on held-out videos of every generator, with five defective and five non-defective target examples in context. A strong diagonal with weak off-diagonal cells would indicate generator-specific artefacts; strong off-diagonal cells would indicate a reusable procedure.

Results. The harness evolved on a generator is the best one for that generator in three of four columns (Happy Horse 0.743, Veo 3.1 0.928, Wan 2.7 0.803; Table 11). The Happy Horse harness also transfers to every other generator (0.743–0.850) and is the best harness on Seedance 2.0. Transfer is not uniform across sources, which is expected with 14 to 54 test videos per column and 10–50 training rows per generator.

Table 9: **One inference, end to end.** A held-out pronunciation query (annotated defective), scored by the reported EvoVE harness. All content is taken from the logged prediction record; transcripts are verbatim.

Stage	What happens
Input	Scripted line: "Many people still don't know that getting started with personal finance is not as hard as you *imagine*!" Generated video with its audio track; ten annotated examples of the category in context.
Route (plan)	The category maps to the speech family, axis A: words spoken versus words scripted. The category entry names four tools and a rule for a swallowed syllable: a scripted syllable missing from *both* transcripts inside an otherwise matching line ⇒ defective, confidence 0.7 (threshold 0.65).
Memory	11,992 characters of error-mode entries for the speech family, including two lessons written in iterations 1–2 about swallowed syllables.
Textbook	34,531 characters: general judging sections, the speech-family chapter, and the reading guide for `asr_funasr`.
Tools	FunASR and Whisper transcribe the line with timestamps; both render the final phrase without one syllable of the scripted word ("as hard as" is heard as "hard"), at about 1.8 s. `pron_name_subclass`, `dual_asr_consensus`, and `pron_claim_gate_v2` report no mismatch, because they target substituted words rather than missing syllables.
Judge	The judge aligns the script with both transcripts, finds the syllable absent from both, and applies the plan's swallowed-syllable check.
Verdict	Defective, confidence 0.7, evidence at 1.81 s. Clause status: mispronunciation checked, none; swallowed syllable observed at 1.81 s; name misreading, no occasion. Correct; 172 s.

Table 10: **Per-category F1 at 5-shot on 5 test query dimensions.** Defective-class F1 on the class-balanced test queries of each training category (n queries), every method with five defective and five non-defective examples per category; Macro is the mean over the five categories.

Method	Face reference (n=20)	Character appearance (n=12)	Pronun-ciation (n=30)	Lip sync (n=26)	Text rendering (n=24)	**Macro**
VLM few-shot	**0.802**	0.500	0.800	0.571	0.692	0.676
GEPA	0.800	**0.833**	0.839	0.828	0.846	0.829
ACE	0.762	0.462	0.857	0.640	0.870	0.718
MCE	0.762	0.667	0.857	0.714	0.870	0.774
Meta-Harness	0.556	0.714	0.714	0.846	0.870	0.740
SkillOpt	0.700	**0.833**	0.759	0.600	0.783	0.735
Alita	0.721	0.545	0.765	0.733	0.783	0.711
EvoVE	0.781	0.800	**0.966**	**0.963**	**0.960**	**0.886**

D.3 Transfer Across Judge Backbones

The loop never inspects the judge's weights, so it should apply to any multimodal judge that can read the delivered media. We rerun the complete protocol with Gemini-3.5-Flash, Seed-2.0-Pro, and the open-weight Qwen3-VL-32B. Each backbone learns its own recipe, calibration, and admission decisions from the same rows; only a media adapter changes. Gemini receives the isolated audio track, Seed hears audio only inside the video, and Qwen3-VL sees speech only through an ASR transcript.

How the backbones evolve. On the support folds, the loop improves every judge along a different route. Gemini admits speech and instruction instruments first and a calibrated motion rule later. Seed gains most on speech ($0.750 \rightarrow 0.909$ out-of-fold), through an instrument that listens to the clip and reads the transcript. Qwen3-VL admits almost nothing: its instruments fix as many errors as they add, and its near-binary error probabilities leave no usable operating point.

Unseen queries. The protocol runs unchanged on all three judges. The gain is largest for Seed ($+0.071$), whose harness improves speech and instruction following, and is small for Gemini and Qwen3-VL. Each backbone is a single run, so these per-judge differences are indicative rather than conclusive.

Data. Each test pair holds two videos generated from the same prompt, one of 15 VBench++ dimensions, and the video raters preferred. *Agreement* is the share of pairs on which an evaluator

Table 11: **Cross-generator transfer.** Each row is the generator on which the harness was evolved (from the same untrained seed) and each column is the generator being evaluated; shaded cells are the harness's own generator. Balanced accuracy, macro-averaged over the two reference-family categories (character identity, reference consistency), with the harness frozen and five defective and five non-defective target examples per category in context. Test videos are plan-disjoint from every support and reward video of every generator, so all cells are held out. n = 24, 14, 22, 54 test videos per column. Best per column in bold.

Source ↓ / Target →	Happy Horse	Seedance 2.0	Veo 3.1	Wan 2.7
Happy Horse	**0.743**	**0.850**	0.831	0.743
Seedance 2.0	0.657	0.725	0.893	0.707
Veo 3.1	0.635	0.725	**0.928**	0.718
Wan 2.7	0.671	0.500	0.866	**0.803**

Table 12: **Judge-backbone transfer.** The same construction protocol is rerun with three judge VLMs; only the judge changes. *Direct*: the judge with the same support annotations and no harness. Micro F1 over the three unseen categories (120 judgments); Δ: paired difference with 95% bootstrap interval. Audio access differs by judge: isolated audio (Gemini), audio inside the video (Seed), ASR transcript only (Qwen).

Judge	**Direct**	**EvoVE**	ΔF1 [95% CI]
Gemini-3.5-Flash	0.754	0.759	$+0.006$ [-0.064, $+0.074$]
Seed-2.0-Pro	0.689	**0.760**	$+0.071$ [-0.014, $+0.161$]
Qwen3-VL-32B (open)	0.662	0.649	-0.014 [-0.052, $+0.023$]

picks the same video; chance is 0.5. We freeze 100 pairs per dimension, 1,500 in total, none sharing a prompt with the adaptation data.

Adaptation. The judge (Gemini-3-Pro) stays frozen. EvoVE learns from 20 labeled pairs per dimension: four serve as text-only examples, eight as fit pairs for the analyst, and eight, never shown to any agent, select instruments and the iteration. The selected harness admitted four instruments and was frozen before any test pair was scored.

E Additional Ablations

This section complements Section 4.3 with one question: is label-gated tool admission necessary?

E.1 Label-Gated Tool Admission

EvoVE admits a synthesised instrument only if, cross-fitted on the ten support labels, it raises out-of-fold F1 over the direct judge without lowering balanced accuracy. Tool-creating agents instead deploy a capability once it executes. The *admission disabled* condition reproduces that rule with the same instruments and calibration. Two controls bracket it: the direct judge, and a prompt-only harness with no instrument. All conditions are scored on the same 120 unseen queries with each of three judges.

Findings. On Seed-2.0-Pro, the gate rejects a motion instrument that would have lowered motion F1, and the gated harness is the best condition on both metrics. On Gemini-3.5-Flash, it matches the other conditions on F1. On Qwen3-VL-32B, whose ten support labels give no instrument an edge, it conservatively admits none. The gate thus prevents harmful admissions without retraining the judge, and its decisions become sharper as more labels are available.

F Trace Analysis

This section gives the material behind Section 5.1: one diagnosis episode, the mapping from findings to mechanisms, and an audit of all traces.

Table 13: **Tool-admission strategies.** Each row replaces the admission rule of EvoVE and keeps everything else fixed: the constructed instruments, the support calibration, the judge, and the 120 unseen queries. BA is balanced accuracy and F1 micro F1 over the three unseen categories. *Tools* gives the instruments deployed and, after the slash, how many of them are non-generalising, i.e. lower the category's unseen-query F1 below the direct judge's. *Admission disabled* deploys every constructed instrument, which is the rule of tool-creating agents that admit on successful execution.

	Gemini-3.5-Flash			Seed-2.0-Pro			Qwen3-VL-32B		
Admission rule	BA	F1	Tools	BA	F1	Tools	BA	F1	Tools
Direct judge (no harness)	0.717	0.754	0/0	0.692	0.689	0/0	0.592	0.662	0/0
Prompt-only harness (no instruments)	0.725	0.772	0/0	0.717	0.712	0/0	0.592	0.675	0/0
Admission disabled (every instrument)	0.742	0.763	3/1	0.750	0.750	3/1	0.667	0.714	3/1
Cross-fitted gate (EvoVE)	0.725	0.759	2/1	0.758	0.760	2/0	0.567	0.649	0/0

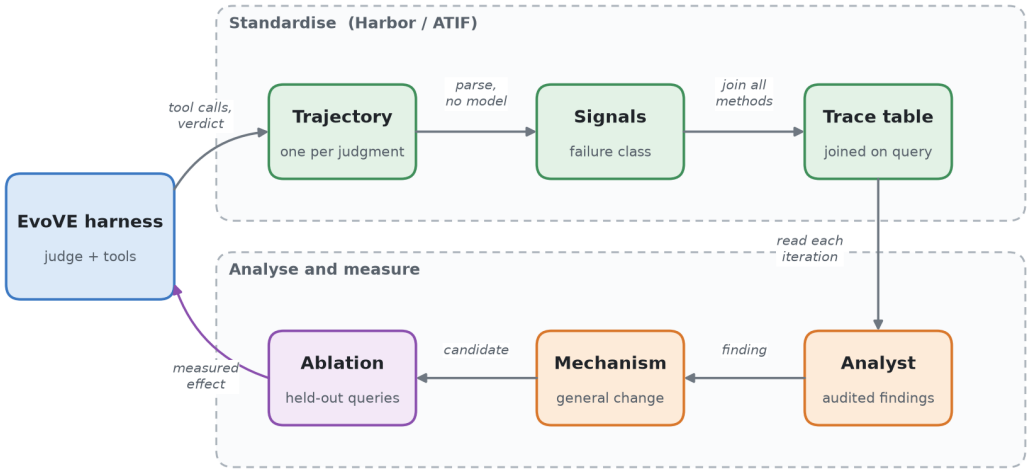


Figure 4: **Trace-based diagnosis.** All judgments are exported in ATIF, converted into deterministic failure signals, and compared by query. Trace-derived mechanisms are accepted only after held-out ablation.

F.1 A Diagnosis Episode: the Routing Gap

Table 14 traces the largest per-category gain of test-time adaptation to the operator outputs that produced it. No test label is used at any point.

Step 1: the analyst audits the route. The transferred plan had no entry for motion and physics, so this family fell back to a default route: one pass over the video, with no plan, frames, probes, or tools. On the ten support examples this route found one of five defects (F1 0.333). All four misses carried the same unverified acquittal the judge wrote on the correct negatives. Every missed defect lasted under one second, too short for one frame per second to resolve.

Step 2: the analyst finds an unrouted signal. The frozen tool bank already contained `clip_drift`, admitted for reference consistency but never shown to the judge for this family. Its mean frame-to-frame similarity separated the support examples completely (defective 0.868–0.936, non-defective 0.941–0.989). Its largest-change timestamps matched the missed defects. A specificity check showed the signal is near chance on other unseen categories, so it tracks motion errors, not generally worse videos.

Step 3: the planner writes a route. The new route reads the `clip_drift` values first, builds a ledger of every interaction in the prompt, and probes frames around the largest changes. A negative

Table 14: **Motion and physics during test-time adaptation.** Each row is one state of the harness on the unseen category *motion & physics*. Support F1 is measured leave-one-out on the ten annotated examples; test F1 on the 26 test queries, which were never shown to any operator. *Gate*: the continuity rule written by the planner on `clip_drift` (mean consecutive-frame similarity < 0.940, minimum < 0.80, or similarity to the first frame < 0.70).

Harness state	What the judge received for this category	Support F1	Test F1
Transferred, examples in context	global default route: one pass over the video, no plan, no frames, no tools, generic textbook sections	0.333	0.538
One adaptation pass	family route written (agent loop with frame and zoom probes, five-clause checklist), no instrument	–	0.500
Adaptation round 2–3	family route + `clip_drift` continuity gate, silent video, 16 frames at 1280 px, interaction ledger, probes at the largest frame-to-frame changes	0.800–0.909	**0.714**

verdict must cite the gate values and one verified interaction. Support F1 rose to 0.909, and test F1 rose from 0.538 to 0.714, above VLM few-shot (0.686) and MCE (0.640). The threshold chosen on ten examples kept its separation on the 26 test queries.

Guarding against over-fitted rules. Ten examples can suggest rules that conflict with the annotators' taxonomy. The traces exposed two such rules for instruction following, which motivated the requirement below that every rule quote the definition clause it implements.

Fixes derived from the traces. The same analysis found a runtime defect: memory serving dropped every entry nested under a matching section, so the motion judge received 469 of 72,056 memory characters. We apply three corrections and re-score every configuration: memory serving keeps sub-sections, each unseen category's examples are split into an analysis half and a hidden selection half, and every rule must quote the definition clause it operationalises.

F.2 From Findings to Mechanisms

Table 15 lists each verified trace finding and the mechanism the loop runs by default in response. Each mechanism can be switched off for ablation.

F.3 The Audit at Scale

We exported every configuration's trajectories in one schema: 5,093 verdicts, including 2,788 EvoVE trajectories with 13,828 probes. Seven slices were each analysed by one agent and independently recomputed by an adversarial verifier. Of about 590 numerical claims, 75% were confirmed and about 40 conclusions were deleted. Every surviving claim passed a base-rate check against constant predictors.

Misses are a criterion problem, not an evidence problem. In most misses, the judge's nearest probe lies within one second of the defect: it looks in the right place. What remains is the criterion it applies, for example a planner checklist that covered only part of the category definition. This finding motivated the coverage guard in Table 15, which rejects any plan that leaves a definition clause without a path to a defective verdict.

Gates are routers, not terminators. Scope exits are reliable: a non-defective verdict that exits on "no reference material" is wrong in 0 of 65 samples. The unreliable exits are "checklist clean" (43.8% wrong) and unspecific ones (44.4%). Deleting the gates would break 285 correct exits; a second stage that can overturn them is the repair the data support.

Probing trades recall for precision. Configurations that probe the video raise precision: their false-alarm rate falls from 0.425 to 0.226 at matched balanced accuracy. Probe count alone does not predict correctness, so the harness uses traces to decide which verdicts deserve a second look.

Table 15: **From verified trace findings to harness mechanisms.** Each finding of the trace audit is answered by a mechanism the loop runs by default (a flag turns it off for ablation); the last column gives its measured effect on the unseen categories.

Finding (traces)	**Mechanism in the loop**	**Effect**
Plan rules narrowed the definition; two clauses had no conviction path	Coverage guard: the planner's output is rejected if a definition clause loses its conviction path; the audit is written for the next analyst	pending (guarded round blocked by the judge gateway)
Gate exits ended inspection before clauses were checked (18 of 68 errors)	Clause check: the judge returns a status per clause; a missing clause sends the verdict back once	part of row (i): $+0.098$ with 36 fixed / 37 broken
145 of 220 verdicts pinned at "non-defective, confidence ≤ 0.7", 53 of them misses	Second stage: a plan-free, definition-anchored reviewer re-examines them and overturns with confidence ≥ 0.8	$+0.139$ [$+0.058$, $+0.223$]
Memory budget set to 0 by the planner; oldest-first truncation served stale entries	Ranked memory with a floor: category, family, general, newest first; a route budget below the floor is raised	-0.042; helps speech, hurts visual
Coverage tables cited frames never fetched (52 of 80 visual tables)	Receipt check: a verdict citing unfetched times is sent back with two probe rounds	-0.055; receipts fixed, verdicts unchanged
Plan branched on the generator; folds were 7/3 by generator	No generator rules; folds stratified by generator and class	protocol
Plan grew from 97 to 212 KB while recall fell	Served plan text capped at 45k characters per category; merge or delete, never add a rule per failed row	protocol
The same harness scored 1.000 and 0.824 on a 15-query fold an hour apart	Selection margin: a round replaces the incumbent only by more than the fold noise; ties keep the smaller harness	protocol
Reward-set refinement scored 0.976 in-reward and 0.439 held-out	samples identifiers hidden from every operator; selection on the held-out fold only	protocol

Benchmark artefacts. Requested duration, generator, and rendered duration are collinear in the earlier benchmark, so retiming effects cannot be isolated. The speech category boundary is consistent across annotators ($\kappa = 0.93$ on 40 relabeled samples) but cannot be resolved from a trace. Eight queries that no configuration gets right are queued for re-annotation; none is treated as a label error.