TEMPO-ZERO: RECURSIVE SELF-PLAY IN EXECUTABLE WORLDS FOR VIDEO REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Video reasoning requires understanding how events unfold over time, yet videos with accurate temporal annotations are expensive to collect. Self-play offers an alternative by letting a model generate and solve its own tasks without manual annotation. However, existing video self-play methods train on fixed sets of recorded videos, so they cannot alter the underlying event dynamics and rely on noisy pseudo-labels derived from the model's own outputs, such as majority voting, that risk reinforcing shared model errors. To overcome these limitations, we introduce Tempo-Zero, a self-improvement framework that closes a loop of *executable environments*, *execution-verified self-play*, and *environment evolution*, where executing a program renders a video with the ground-truth option and interval of every question. A single policy plays both Questioner and Solver, proposing questions and answering them with timestamped evidence lines, and is trained on execution-grounded rewards instead of model consensus. The Solver's performance steers a curriculum over tasks and difficulty levels, and the policy selects rule changes that enrich the event dynamics of later rounds. Despite being trained solely on simple programs, without supervised fine-tuning or external video data which prior methods typically require, Tempo-Zero outperforms the base models and recent self-play baselines across twelve real-world video benchmarks at different model scales, with particularly strong gains on temporal grounding tasks. Code will be made publicly available.

1 INTRODUCTION

Video understanding and reasoning require models to interpret dynamic events, track temporal dependencies, and perform multi-step inference over time (Chen et al., 2025; Rasheed et al., 2026; Hu et al., 2026). While supervised fine-tuning and reinforcement learning for vision-language models have improved these capabilities (Bai et al., 2025; Wang et al., 2025; Chen et al., 2025), progress remains constrained by the cost of collecting diverse videos, designing multi-step tasks, and verifying temporal annotations (Rasheed et al., 2026; Huang et al., 2024). To reduce dependence on human annotation, self-improvement has emerged as a promising alternative paradigm, enabling models to construct their own training tasks and improve through self-generated feedback (Zou et al., 2026; Zhao et al., 2025a; Huang et al., 2026a; Liu et al., 2025a).

Self-play provides a practical framework for self-improvement: a model proposes tasks, solves them, and updates its policy based on the outcomes of its attempts (Silver et al., 2017; Chen et al., 2024; Bailey et al., 2026). In text, it has improved coding and mathematical reasoning, with answers verified either through programmatic execution (Zhao et al., 2025a), by majority voting over sampled solutions (Huang et al., 2026a), or against source documents (Liu et al., 2025a). Beyond text, image self-play derives answer labels either through majority voting (He et al., 2026a; Wang et al., 2026a) or from the outcomes of visual games executed on rendered and edited images (Wang et al., 2026c), yet static images lack temporal dependencies and continuously evolving states. Extending this recipe to video, frameworks such as Video-Zero (Zhang et al., 2026b) and EvoVid (Huang et al., 2026b) train on fixed collections of pre-recorded, unannotated videos, which introduces two limitations. First, because these videos lack verifiable ground truth, answers are labeled from the model's own outputs, for example by majority voting over responses sampled from the same model (Figure 1a), which risks reinforcing systematic errors shared across samples and propagating them through training. Second,

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.

the model can vary its questions but not the events it observes. Consequently, the event dynamics and temporal complexity of its training data remain fixed even as its reasoning improves.

Executable programs offer a foundation for verifiable and controllable video self-play. In code self-play, executing a model-proposed program on an input yields an automatically verifiable answer (Zhao et al., 2025a). For videos, an executable program produces both the visual input and its event trace: its entities and transition rules determine the event dynamics in the rendered video, while task templates construct questions from the recorded events. The trace therefore provides each question's correct answer option and the exact time interval of the evidence needed to answer it (Yi et al., 2020; Jiang et al., 2026). Thus, programs control both the videos shown to the model and the ground-truth supervision used to train it, and re-executing a program under changed rules updates both. Although program-generated videos already serve as training data, programs designed before training fix the templates, task allocation, difficulty, and event dynamics (Brown et al., 2025; Jiang et al., 2026). The resulting training distribution cannot adapt to the learner's changing strengths and weaknesses, limiting its exposure to appropriately challenging temporal structures needed for continued improvement (Figure 1b). Self-play on recorded videos thus adapts only its questions and cannot verify its labels, whereas program-generated video has verifiable labels but adapts neither its questions nor its events to the learner.

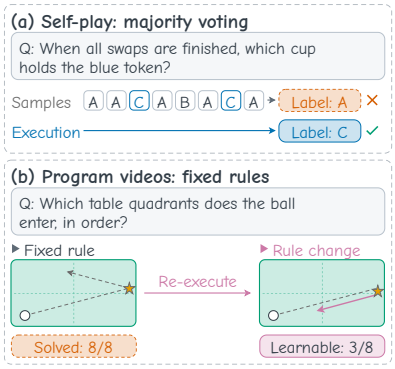


Figure 1: Limitations of video self-play and program-generated video. (a) On a shell-game video, majority voting over eight sampled answers selects a wrong label over the two correct ones, while execution gives the correct label. (b) Under a fixed rule, all Solver rollouts are correct; changing the cushion rule and re-executing the program yields a new path with exact labels.

Based on these observations, we introduce Tempo-Zero, which turns executable programs into an evolving source of video experiences and verifiable supervision. The framework has three components. First, *executable environments* execute programs to render videos and produce their event traces. Each trace yields a set of verifiable criteria, each specifying a question, its correct answer, and the time interval of the event that determines that answer. To ensure that this supervision is available to a frame-based video model, we recompute each criterion on the sampled-frame timeline and discard it if its answer differs from the execution label. We also screen for temporal dependence by recomputing each criterion under reversed and permuted event order, removing criteria whose answers remain unchanged under both evaluable transformations. The retained criteria therefore describe events that the sampled frames show. Second, *execution-verified self-play* uses a single shared vision-language policy as both Questioner and Solver. The Questioner proposes questions linked to verified criteria, and the Solver answers them with timestamped evidence lines. Execution rewards the Questioner for valid questions near the Solver's current capability and rewards the Solver for the correct answer, granting interval credit only when a cited line states the correct event, rather than relying on model consensus. Finally, *environment evolution* adapts not only which tasks are sampled but also what can happen in the videos. Pass rates reallocate training across tasks and difficulty levels, and the policy selects validated rule changes, whose re-execution generates new event dynamics together with recomputed answers and evidence intervals. Tempo-Zero therefore moves beyond asking different questions about fixed videos: it creates progressively targeted video worlds whose supervision stays exact.

Our main contributions are summarized as follows:

- **Executable environments for video self-play.** We formalize video task generation as program execution, where rule changes produce new event dynamics with ground-truth options and intervals, without recorded videos or manual annotations (§3.1).
- **Execution-verified self-play with environment evolution.** One policy plays Questioner and Solver and learns from execution-grounded rewards instead of majority voting, while a pass-rate curriculum and policy-selected rule changes evolve the environments with the learner (§3.2–3.3).
- **Performance gains on real-world videos.** Trained solely on rendered videos, without supervised fine-tuning, Tempo-Zero improves over the base models across twelve real-world video bench-

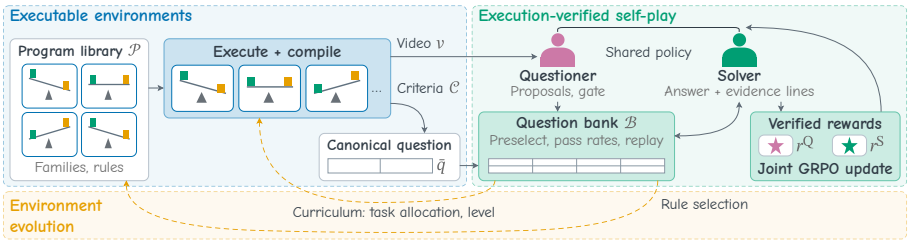


Figure 2: Overview of Tempo-Zero. Programs in library $\mathcal{P}$ execute and compile into video v, criteria $\mathcal{C}$, and canonical question $\bar{q}$. One policy proposes questions as the Questioner and answers them as the Solver; bank $\mathcal{B}$ stores measured questions for replay, and execution-grounded rewards r^{Q} and r^{S} drive GRPO updates.

marks on four backbones from two model families and outperforms recent self-play baselines, especially in temporal grounding (§4).

2 Related Work

Self-play. Self-play has evolved from optimizing policies on self-generated responses to constructing tasks and evaluation signals autonomously. Early text methods contrast a model's answers with reference demonstrations (Chen et al., 2024) or train it to score its own outputs (Yuan et al., 2024). Later frameworks split self-play into a Questioner and a Solver with feedback from code execution (Zhao et al., 2025a), majority voting (Huang et al., 2026a), text corpora (Liu et al., 2025a), or a guide model (Bailey et al., 2026), and self-improving agents modify their own code (Zhang et al., 2026a; Wang et al., 2026d) or co-evolve their evaluators (Iacob et al., 2026). In the visual domain, image self-play labels answers by majority voting (He et al., 2026a; Wang et al., 2026a) or by visual games (Wang et al., 2026c). Video self-play then brings time into the loop: Video-Zero adds temporal evidence to question answering (Zhang et al., 2026b), EvoVid rewards questions whose Solver confidence drops when frames are permuted (Huang et al., 2026b), and video reinforcement learning rewards temporal order (Feng et al., 2025; He et al., 2026b) or cited evidence (Meng et al., 2026; Liu et al., 2026b). These video methods, however, train on fixed collections of recorded videos and take their labels and intervals from annotations, model consensus, or the position of a sampled clip. Tempo-Zero removes both dependencies: it generates the videos from executable programs, so every label, including the time span of each event, is computed by execution rather than by model consensus, and its temporal checks screen these labels before any rollout.

Programmatic environments. Programmatic synthesis couples simulation with deterministic supervision and turns execution traces into verifiable ground truth. For video, SynRL synthesizes videos with automated temporal supervision (Jiang et al., 2026), and SIMS-V (Brown et al., 2025) and SpatialEvo (Li et al., 2026a) turn 3D simulation into spatial question-answering targets. Executable code also represents worlds for physical reasoning (Wang et al., 2026b) and video generation (Li et al., 2026b). In SynRL and SIMS-V, however, the rules and question templates are fixed before training, so the data cannot follow the learner. Adaptive curricula address this gap: ACCEL mutates environment parameters (Parker-Holder et al., 2022), Prioritized Level Replay revisits levels with high learning potential (Jiang et al., 2021), and SPADE trains agents to construct executable environments (Liu et al., 2026a). Tempo-Zero brings this adaptation to video self-play and closes the loop between the learner and its environments. Its pass rates steer the curriculum, the policy itself selects rule changes, and re-execution recomputes every video and label. The training distribution thus grows with the learner, and its supervision stays exact.

3 Method

Tempo-Zero trains a vision-language policy by self-play in an executable program library, where every label comes from program execution rather than majority voting (Figure 2). Programs first execute into rendered videos and verification criteria (§3.1). One policy then proposes and answers questions, and the execution trace scores both roles (§3.2). Finally, a pass-rate curriculum and policy-selected rule changes evolve the environments with the learner (§3.3).

Problem formulation. We formalize video self-improvement as an interaction between two roles of a single policy π_θ: a Questioner π_θ^{Q} that proposes questions and a Solver π_θ^{S} that answers them.

The two roles share parameters θ and differ only in their prompts, so visual representations learned while solving also inform question generation. Executing a program from library $\mathcal{P}$ yields a synthetic video v with verification criteria $\mathcal{C}$, each recording a question stem, its ground-truth option, and the time interval of the event that decides it. With the criteria as the source of all rewards, the policy is trained to maximize

$$J(\theta) = \mathbb{E}_w \, \mathbb{E}_{q \sim \pi_\theta^{\mathrm{Q}}(\cdot \mid w)} \, r^{\mathrm{Q}}(q) \; + \; \mathbb{E}_{q \in \mathcal{Q},\, y \sim \pi_\theta^{\mathrm{S}}(\cdot \mid v_q, q)} \, r^{\mathrm{S}}(y, q), \tag{1}$$

where instance w is drawn by the curriculum (§3.3), $\mathcal{Q}$ holds the questions that train the Solver, namely the compiled canonical question of each drawn instance, at most one proposal per instance, and replayed bank questions (§3.2), and v_q is the video of question q. Questioner reward r^{Q} prioritizes valid questions near the limit of the Solver's current capability, and Solver reward r^{S} scores the answer together with how well its cited evidence matches the execution trace. Answered questions enter bank $\mathcal{B}$ with their pass rates, the fraction of Solver rollouts that choose the ground-truth option.

3.1 Executable Environments

Executable environments compute labels with the same process that produces the video.

Programs and instances. Library $\mathcal{P}$ groups programs into environment families f. Each family defines physical entities, state-transition rules with alternative values, and the tasks it supports. Within a family, an instance w is set by three quantities: difficulty level ℓ scales entity counts and operation complexity, rule configuration u_f selects the active transition behaviors, and playback scale λ_r sets video duration. Instance parameters x fix the remaining randomness (Appendix B). Given these inputs, program p_f executes into trace τ, which compiles into video v, criteria $\mathcal{C}$, and canonical question $\bar{q}$ (Figure 3):

$$\tau = \mathrm{Exec}(p_f;\, \ell, u_f, x), \qquad (v, \mathcal{C}, \bar{q}) = \mathrm{Compile}(\tau;\, \lambda_r). \tag{2}$$

Because v and $\mathcal{C}$ come from the same trace, changing ℓ or u_f changes the event dynamics and the ground truth together. Compile writes $\bar{q}$ for the task allocated to the instance, using the stem and ground-truth option of the matching criterion and distractors from its domain.

Criteria and tasks. Execution traces yield reliable supervision only when each question is tied to a dynamic event that the policy can see. Each learning target is therefore formalized as criterion $c \in \mathcal{C}$. A criterion specifies a question stem, ground-truth option $y^\star$, ground-truth interval $e_c^\star$ (one or more time segments), and a distractor domain with a minimum resolution floor. An execution label replaces model consensus only if it holds on the frames the policy sees and its answer depends on event order, so each criterion passes two programmatic checks before training.

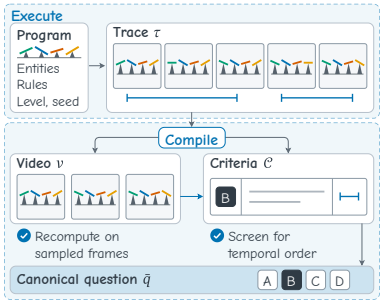


Figure 3: Environment compilation. Ground truth is recomputed on the sampled frames, and criteria are screened for temporal order. Colors identify entities, blue brackets mark ground-truth intervals, and the dark option is the ground-truth option.

Frame recomputation. Programs advance state on a fine simulation clock, while the policy observes only frames sampled at a much lower rate, so when an event starts or ends between two sampled frames, a label read from the simulation clock describes a state that no sampled frame shows. We therefore express every temporal label on the timestamps of the sampled frames and recompute each ground-truth value from those frames. A criterion is discarded when its value cannot be recomputed from those frames or disagrees with the simulation (Appendix B.3).

Static-question screen. A question whose answer does not depend on the order of events is static. To detect such questions, we recompute each criterion's answer on the reversed trace and on a trace with permuted events, and we exclude criteria whose answer stays the same under both transformations. When a transformation yields no determinate answer, the screen keeps the criterion, which must still show its target in several sampled frames (Appendix B.2).

After both checks, the retained criteria span the tasks in Table 6, such as event count, event order, and state change. Each family supports a subset of these tasks, and the (family, task) pair (f, a) serves as the scheduling unit of the curriculum (§3.3).

3.2 EXECUTION-VERIFIED SELF-PLAY

Given compiled instances, self-play needs a validity check for each proposed question and a reward for each answer. Tempo-Zero computes both from the execution trace, so the loop involves no judge model or majority voting (Figures 1a and 2). In each step, the Questioner proposes questions, a programmatic gate filters them, the Solver answers them with timestamped evidence lines, and bank $\mathcal{B}$ keeps questions for replay (Figure 4).

Question proposals. Each compiled instance already carries its canonical question $\bar{q}$, which the Solver answers whether or not any proposal passes the gate. The Questioner asks about the same verified events in new ways: it selects one of the criteria and writes a new stem and distractors, so the policy adapts its questions to the Solver's current capability while the criterion fixes the ground truth. Because the stem and options are free-form, they can be ambiguous or invalid, or leak the answer. To write a proposal, the Questioner receives the video, execution metadata listing events and their timestamps, the askable criteria in $\mathcal{C}$, same-task questions from the bank, and the Solver's pass rate per task (Appendix B.4). Its output, a proposal q, names one criterion $c_q \in \mathcal{C}$ and contains a question stem, the ground-truth option $y^\star$ of c_q, distractors, and evidence lines citing the events that the criterion depends on. Because the metadata already provides verified timestamps, a localization reward would only credit copying them, and the Questioner therefore receives none. Each proposal then passes through a programmatic gate $g(q) \in \{0, 1\}$ that checks format, option validity, task scope, and answer leakage in the stem. Option validity requires exactly one option to equal $y^\star$ and every distractor to be separated from $y^\star$ and from the other distractors by the resolution floor. For accepted proposals, the environment shuffles the options uniformly at random and assigns option letters; proposals with $g = 0$ are not answered.

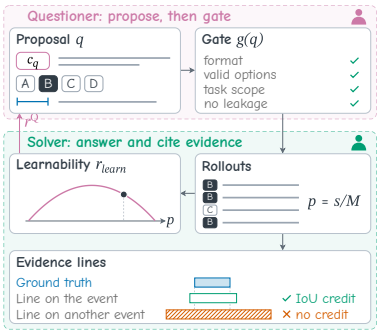


Figure 4: Execution-verified self-play. The Questioner's proposal is bound to criterion c_q and passes gate $g(q)$. The Solver's rollouts give pass rate p, which sets r_{learn} for the Questioner, and only an evidence line that states the criterion's event earns interval credit.

Question bank. Bank $\mathcal{B}$ serves three purposes: it supplies same-task examples and per-task pass rates to the Questioner, replay questions to the Solver, and pass-rate statistics to the curriculum (§3.3). Among the gated proposals, it preselects at most one per instance for Solver training, favoring tasks that are rare in the bank, and stores it with each measured canonical question. When the bank reaches capacity, eviction first removes questions of overrepresented tasks and families and then the oldest measurements, so all twelve tasks stay represented.

Answers and evidence. On the Solver side, an answer letter alone does not show whether the Solver located the events that decide it, since multiple-choice answers can often be inferred from isolated frames. The Solver therefore cites its evidence. It receives only the video, the question, and the options, and it answers every question with M rollouts under π_θ^{S}. Across the M rollouts, pass rate $p(q) = s/M$, where s rollouts choose $y^\star$, measures question difficulty. Rollouts on every gated proposal score the Questioner, and those on canonical questions, preselected proposals, and replayed questions also train the Solver. Each rollout y contains an answer letter, whose option we denote $\hat{y}$, and L evidence lines (I_k, d_k), each pairing a predicted interval I_k with an event statement d_k. The evidence lines are scored against the criteria $\mathcal{C}_q \subseteq \mathcal{C}$ of question q, namely c_q and the criteria of the events its answer rests on. The program parses the lines with a fixed grammar and assigns them one-to-one to these criteria: line $k(c)$ is assigned to criterion c, and $\delta_c \in \{0, 1\}$ records whether it states the event of c, that is, asserts the recorded value of c about its subject (Appendix C.1). With m matched criteria, evidence score r_{evid} is the statement-level F_1 score, with precision m/L and recall $m/|\mathcal{C}_q|$. Grounded score r_{grd} averages the temporal Intersection over Union (IoU) over the criteria and counts only matched lines:

$$r_{\text{evid}} = \frac{2m}{L + |\mathcal{C}_q|}, \qquad r_{\text{grd}} = \operatorname*{mean}_{c \in \mathcal{C}_q} \delta_c \operatorname{IoU}\big(I_{k(c)}, e_c^\star\big). \tag{3}$$

A simpler grounding reward, $\max_k \operatorname{IoU}(I_k, e_c^\star)$, would credit a broad interval on any line, even one that describes another event (Figure 4); r_{grd} instead counts overlap only on the line whose statement matches the criterion, so interval credit requires the correct event.

Reward terms. Both rewards combine these terms and are computed programmatically from the execution trace, the Solver's rollouts, and the bank:

$$r^{\mathrm{S}}(y,q) = r_{\mathrm{ans}} + \mathrm{mean}\big(r_{\mathrm{evid}}, r_{\mathrm{grd}}\big), \qquad r^{\mathrm{Q}}(q) = g(q)\ \mathrm{mean}\big(r_{\mathrm{learn}}, r_{\mathrm{nov}}\big). \tag{4}$$

Here $r_{\mathrm{ans}} = \mathbf{1}[\hat{y} = y^{\star}]$ is the answer reward, and each term lies in $[0,1]$, so $r^{\mathrm{S}} \in [0,2]$. The Solver's answer and evidence are rewarded separately, so a rollout that cites the correct event earns evidence credit even when its final choice is wrong, while r_{ans} still separates correct from incorrect answers. For the Questioner, gate $g(q)$ removes the reward of invalid proposals, and learnability reward r_{learn} favors questions that the Solver neither always solves nor always misses. It is the variance of a Bernoulli variable whose success probability is the Laplace-smoothed pass rate $\tilde{p}$, scaled to peak at one:

$$r_{\mathrm{learn}}(q) = 4\,\tilde{p}\,(1-\tilde{p}), \qquad \tilde{p} = \frac{s+1}{M+2}. \tag{5}$$

Smoothing keeps the reward positive when all rollouts are correct or all are wrong. Finally, novelty reward $r_{\mathrm{nov}} \in [0,1]$ decreases with the number of bank questions that share a proposal's task signature, which keeps the Questioner from repeating one question pattern (Appendix C).

Policy update. Both rewards feed a single joint update of the shared parameters θ. Within step t, proposals, Solver rollouts, and both rewards are computed under the same snapshot π_{θ_t} and held fixed during the update, so each Questioner reward reflects the current Solver and passes no gradient through its pass rate. The policy is then updated in one Group Relative Policy Optimization (GRPO) step (Shao et al., 2024) over the groups of both roles. A Solver group holds the M rollouts of one question, and a Questioner group holds the proposals of one instance. Within each group $\mathcal{G}$, the advantage of sample i is its mean-centered reward, $\hat{A}_i = r_i - \bar{r}_{\mathcal{G}}$, where $\bar{r}_{\mathcal{G}}$ is the mean reward of $\mathcal{G}$. Groups whose rewards are all equal carry no advantage and are omitted from the update.

3.3 Environment Evolution

Fixed programs freeze the task allocation, difficulty, and event dynamics of training (Figure 1b). As the policy improves, more Solver groups agree, so the answer reward no longer separates their rollouts. Tempo-Zero therefore changes the environments in two ways (Figure 5): the curriculum reallocates instances across (family, task) pairs and levels by pass rate, and the policy selects rule changes that the executor validates.

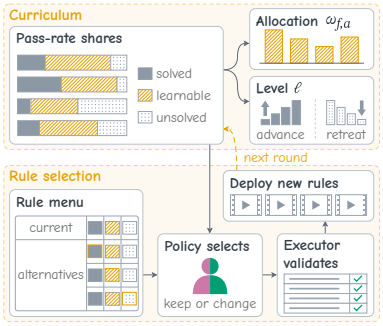


Figure 5: Environment evolution. Pass rates guide allocation $\omega_{f,a}$ (Eq. 6) and set level ℓ, and the policy selects a rule change, which the executor validates before deployment.

Task allocation. Uniform instance sampling spends updates on tasks that the policy already solves or cannot yet solve. To avoid this, the curriculum reads exponentially decayed pass-rate statistics in bank $\mathcal{B}$ after each round and fixes the allocation and levels of the next round. Each family receives an equal share of instances, and within family f, pair (f,a) receives

$$\omega_{f,a} = \kappa/n_f + (1-\kappa)\,w_{f,a}\big/\textstyle\sum_{a'} w_{f,a'}, \tag{6}$$

where coverage share κ spreads part of the quota evenly over the n_f tasks of f so that every task stays measured, and the policy sets task weight $w_{f,a}$ from pass rates. The same statistics move difficulty: a pair's level ℓ rises when its decayed pass rate reaches ρ_{adv} and falls at ρ_{ret}.

Rule selection. Allocation redistributes the existing environments but cannot create new event dynamics, which requires changing the rules. Rules decide which dynamics a family can produce, and Tempo-Zero lets the policy choose how they change. After each round except the last, the environment runtime composes a rule menu for every family. The menu lists the current configuration and alternatives that each move one rule, or in later rounds a pair of rules, to values the family has not run. The policy reads the family's pass rates and recorded responses under the current configuration, sets the task weights $w_{f,a}$ of Eq. 6, and either keeps u_f or selects a menu entry, optionally with operation-graph edits (Appendix B.5). Before deployment, the executor checks each selected change for validity and re-executes the program to preview the resulting videos and criteria; an invalid change keeps the previous program. Rule selection runs at inference and receives no gradient update. Independently of rule selection, playback scale λ_r follows a fixed schedule that lengthens videos across rounds r. Algorithm 1 in Appendix A.1 summarizes the full training loop.

Table 1: Temporal grounding on real-world benchmarks. Metrics in %; Avg. averages the mIoU of the first three benchmarks and GQA@0.3. Bold and underline mark the best and second-best per column at 4B and 8B; arrows give the change over the base model, and other methods have no 30B-A3B or Gemma results.

	Method	ANet-RTL				ActivityNet Captions				Charades-STA				NExT-GQA		Avg.
		mIoU	R@0.3	R@0.5	R@0.7	mIoU	R@0.3	R@0.5	R@0.7	mIoU	R@0.3	R@0.5	R@0.7	GQA@0.3	GQA@0.5	
4B	Base model	16.86	22.70	15.72	8.73	22.86	30.94	19.04	11.29	37.52	58.90	36.12	16.23	23.09	15.40	25.08
	VisPlay	16.58	23.14	13.97	8.73	22.89	30.98	19.12	11.31	37.59	58.82	36.59	16.42	23.14	15.33	25.05
	V-Zero	15.85	20.96	13.97	8.30	22.81	30.91	18.92	11.31	37.10	58.33	35.24	15.62	22.82	15.22	24.64
	Video-Zero	29.75	44.98	28.82	**13.54**	32.12	44.76	27.67	16.24	41.85	65.11	40.81	**20.03**	38.29	23.00	35.50
	Tempo-Zero	**31.34** ↑14.48	**48.03** ↑25.33	**31.00** ↑15.28	13.10 ↑4.37	**34.98** ↑12.12	**49.35** ↑18.41	**31.04** ↑12.00	**17.26** ↑5.97	**42.53** ↑5.01	**66.29** ↑7.39	**41.26** ↑5.14	19.49 ↑3.26	**39.17** ↑16.08	**24.50** ↑9.10	**37.00** ↑11.92
8B	Base model	21.95	30.57	21.83	10.04	26.56	36.59	22.63	12.85	36.70	59.11	33.87	13.49	35.93	21.83	30.28
	VisPlay	21.83	32.31	20.09	10.04	26.74	36.94	22.99	12.93	36.65	59.03	33.92	13.47	35.87	22.19	30.27
	V-Zero	22.77	32.31	20.52	9.17	26.69	36.79	23.05	13.00	36.43	58.63	33.66	13.41	35.73	22.11	30.40
	Video-Zero	26.19	39.30	24.45	12.23	29.87	41.77	26.23	14.66	37.34	60.16	35.54	13.58	39.60	24.26	33.25
	Tempo-Zero	**31.22** ↑9.27	**46.72** ↑16.15	**28.38** ↑6.55	**13.54** ↑3.50	**34.16** ↑7.60	**49.20** ↑12.61	**30.86** ↑8.23	**16.26** ↑3.41	**46.10** ↑9.40	**72.26** ↑13.15	**49.81** ↑15.94	**23.06** ↑9.57	**40.39** ↑4.46	**25.14** ↑3.31	**37.97** ↑7.69
30B-A3B	Base model	23.45	32.75	23.14	13.10	27.90	37.65	23.98	14.39	47.34	68.44	51.61	30.13	38.83	23.63	34.38
	Tempo-Zero	34.23 ↑10.78	49.78 ↑17.03	33.62 ↑10.48	17.90 ↑4.80	36.35 ↑8.45	52.42 ↑14.77	32.95 ↑8.97	17.38 ↑2.99	48.77 ↑1.43	77.15 ↑8.71	52.96 ↑1.35	22.37 ↓7.76	45.15 ↑6.32	27.88 ↑4.25	41.12 ↑6.74
Gemma	Base model	23.97	34.93	22.27	10.04	18.64	24.93	14.65	7.05	19.08	27.39	13.04	6.05	21.72	10.99	20.85
	Tempo-Zero	30.52 ↑6.55	41.48 ↑6.55	27.51 ↑5.24	15.72 ↑5.68	33.53 ↑14.89	47.52 ↑22.59	29.94 ↑15.29	15.99 ↑8.94	32.01 ↑12.93	50.97 ↑23.58	24.30 ↑11.26	9.52 ↑3.47	26.40 ↑4.68	14.64 ↑3.65	30.61 ↑9.76

4 EXPERIMENTS

We test whether execution-verified self-play improves real-video grounding, long-video understanding, video reasoning, and other architectures.

4.1 EXPERIMENTAL SETUP

Models and training. We train Qwen3-VL-4B-Instruct and Qwen3-VL-8B-Instruct (Bai et al., 2025), each serving as both Questioner and Solver, with LoRA adapters for five rounds on videos rendered by the executable environments alone, without external videos, human annotations, or supervised fine-tuning (Appendix A). Baselines are the instruction-tuned base model and three self-improvement methods: VisPlay (He et al., 2026a) and V-Zero (Wang et al., 2026a), which train on images, and Video-Zero (Zhang et al., 2026b), which trains on recorded videos; all results follow the evaluation protocol of Zhang et al. (2026b).

Benchmarks. The final checkpoint is evaluated zero-shot on twelve real-world benchmarks in three groups. Temporal grounding uses ANet-RTL (Huang et al., 2024), ActivityNet Captions (Krishna et al., 2017), Charades-STA (Gao et al., 2017), and NExT-GQA (Xiao et al., 2024), scored by mean IoU (mIoU), recall at IoU thresholds (R@0.3/0.5/0.7), and grounded QA accuracy (GQA@0.3/0.5). Long-video understanding uses LongVideoBench (Wu et al., 2024), MLVU (Zhou et al., 2025), Video-MME-Long (Fu et al., 2025), and LSDBench (Qu et al., 2025), and video reasoning uses LongVideoReason (Chen et al., 2025), MMVU (Zhao et al., 2025b), VideoMathQA (Rasheed et al., 2026), and VideoMMMU (Hu et al., 2026), both scored by accuracy (Appendix A.4).

4.2 MAIN RESULTS

Temporal grounding. Tempo-Zero improves every grounding metric over the base model at 4B and 8B and gives the best result in 26 of their 28 metrics in Table 1. The margin over Video-Zero widens at 8B: on Charades-STA, Video-Zero gains 0.64 mIoU over the base model and Tempo-Zero gains 9.40. Video-Zero derives the temporal evidence it rewards from the model's own predictions, whereas Tempo-Zero scores evidence lines against intervals computed by execution, which stay exact at any scale (Eq. 3). Training never sees a real-world interval annotation, so these gains reflect localization transferred from rendered to real videos.

Long-video understanding and video reasoning. The gains extend beyond grounding: Tempo-Zero improves all eight benchmarks of Table 2 on every Qwen3-VL backbone and gives the best result in 13 of the 16 metrics at 4B and 8B. Because the training videos contain only simple rendered entities, these results indicate that the temporal skills learned from execution carry over to real-world content.

Other backbones. To test whether the recipe depends on the dense Qwen3-VL backbones, we also train Qwen3-VL-30B-A3B-Instruct, a sparse mixture-of-experts model with 3B active parameters, and Gemma 4 E4B (Gemma Team, 2026), from another model family. In the last two blocks of

Table 2: Long-video understanding and video reasoning. Official accuracy in %; marks as in Table 1.

	Method	Long-Video Understanding				Video Reasoning				Avg.
		LongVideo Bench	Video-MME MLVU	Video-MME Long	LSDBench	LongVideo Reason	MMVU	VideoMathQA	VideoMMMU	
4B	Base model	57.70	62.60	51.60	54.68	69.90	64.32	19.05	57.56	54.68
	VisPlay	59.20	63.50	53.20	54.83	69.60	63.36	18.33	57.44	54.93
	V-Zero	59.50	63.50	51.90	54.68	68.20	64.48	20.95	58.33	55.19
	Video-Zero	59.10	63.40	**53.60**	**56.52**	70.30	64.96	21.67	61.56	56.39
	Tempo-Zero	**61.00** ↑3.30	**67.36** ↑4.76	53.30 ↑1.70	56.06 ↑1.38	**72.40** ↑2.50	**66.08** ↑1.76	**22.38** ↑3.33	**63.11** ↑5.55	**57.71** ↑3.03
8B	Base model	59.80	64.40	55.70	57.59	68.60	68.32	23.81	63.78	57.75
	VisPlay	60.50	64.90	54.90	57.44	68.10	67.04	21.43	62.67	57.12
	V-Zero	60.10	64.20	56.00	56.90	69.20	**68.80**	22.14	63.22	57.57
	Video-Zero	60.40	64.60	56.60	56.75	70.70	68.00	25.48	64.22	58.34
	Tempo-Zero	**61.00** ↑1.20	**68.51** ↑4.11	**56.90** ↑1.20	**58.59** ↑1.00	**73.00** ↑4.40	68.48 ↑0.16	**27.38** ↑3.57	**64.44** ↑0.66	**59.79** ↑2.04
30B-A3B	Base model	65.37	70.26	58.33	62.50	71.40	72.16	26.90	65.11	61.50
	Tempo-Zero	66.70 ↑1.33	71.34 ↑1.08	60.56 ↑2.23	63.96 ↑1.46	75.60 ↑4.20	73.60 ↑1.44	31.90 ↑5.00	67.44 ↑2.33	63.89 ↑2.39
Gemma	Base model	50.71	54.51	46.44	47.09	70.60	65.28	30.00	65.89	53.81
	Tempo-Zero	53.40 ↑2.69	54.62 ↑0.11	48.00 ↑1.56	55.14 ↑8.05	71.10 ↑0.50	67.04 ↑1.76	29.29 ↓0.71	66.11 ↑0.22	55.59 ↑1.78

Tables 1 and 2, Tempo-Zero raises 21 of the 22 metrics at 30B-A3B and 21 on Gemma, so its gains rely on neither a dense architecture nor one model family.

4.3 Ablations

All ablations use Qwen3-VL-4B and remove one component under the same setup (Appendix E.1); the full method is best in every column.

Questioner. The Questioner supplies questions and targets them. *W/o Questioner* trains the Solver only on the canonical questions that the programs compile, and Overall falls from 50.81 to 47.71, the largest Overall drop. Proposals ask new questions about the verified criteria of the same instances, whose ground truth stays fixed, so they widen the training distribution without adding label noise. Targeting comes from the two terms of r^{Q}, of which learnability matters more: *w/o learnability reward* lowers Temporal Grounding by 1.75 points, more than three times the drop of *w/o novelty reward*. Questions at intermediate pass rates yield Solver groups whose rollouts disagree, so the answer reward separates them, whereas novelty only spreads proposals over task signatures.

Table 3: Ablations on Qwen3-VL-4B. Temporal Grounding and Video QA & Reasoning are the Avg. columns of Tables 1 and 2, and Overall averages all twelve benchmarks; marks as in Table 1.

Variant	Temporal Grounding	Video QA & Reasoning	Overall
Tempo-Zero (full)	**37.00**	**57.71**	**50.81**
Questioner			
w/o novelty reward	36.53	56.88	50.10
w/o learnability reward	35.25	57.27	49.93
w/o Questioner	31.86	55.63	47.71
Solver			
w/o evidence score	35.34	57.33	50.00
w/o grounded score	36.63	57.11	50.28
Answer reward only	31.30	56.28	47.96
Environment evolution			
w/o rule selection	35.23	56.58	49.46

Solver. Evidence terms teach the Solver to locate events. With *answer reward only*, $r^{\mathrm{S}} = r_{\mathrm{ans}}$ and evidence lines earn no reward, so Temporal Grounding falls by 5.70 points, the largest drop in its column. Of the two evidence terms, the statement check carries most of this signal. *W/o evidence score* leaves a correct statement credited only through the interval overlap of r_{grd} and lowers Temporal Grounding by 1.66 points, over four times the drop of *w/o grounded score*, whose interval credit refines the statement check. The benefit reaches beyond grounding, since Video QA & Reasoning also drops the most among the Solver variants when only the answer is rewarded.

Environment evolution. Evolution changes which instances are drawn and what happens in them, and *w/o rule selection* removes only the second: it keeps the initial rules, while the curriculum still reallocates instances and the playback schedule still lengthens videos. Temporal Grounding drops by 1.77 points, so reallocation over fixed rules does not supply the event dynamics that rule changes add.

4.4 Analysis

Training dynamics. Trained on execution-grounded rewards, both roles improve over the rounds of the 4B and 8B runs (Appendix D). On 128 held-out instances (Appendix D.2), Solver accuracy

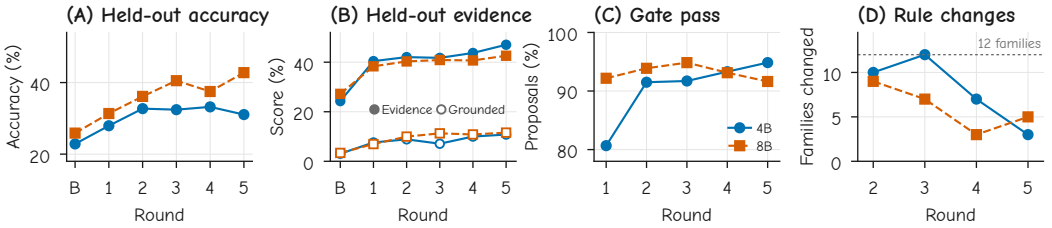


Figure 6: Training dynamics. (A) Held-out accuracy. (B) Evidence (filled) and grounded (hollow) scores; B is the base model. (C) Proposals passing the gate. (D) Families whose rules change for each round.

rises from 25.9% to 42.8% at 8B, and the evidence score nearly doubles at 4B, from 24.3% to 47.0% (Figure 6A–B). The Questioner's gate pass rate rises from 80.7% to 94.9% at 4B and stays above 91% at 8B (Figure 6C), so the policy learns the gate's validity checks. Rule selection is most active early: the policy changes rules in 56 of its 96 family decisions, most of them for rounds 2 and 3 (Figure 6D), and every change stays verifiable, since all menu alternatives compile into instances with recomputed criteria (Appendix D.3).

Execution labels versus majority voting. These gains rest on execution labels, which can replace consensus only if they hold on the frames that the policy sees, as the checks confirm: recomputation discards fewer than 1% of the compiled criteria (Appendix B.3), and every Solver group is scored against a criterion that passed both checks, so its label matches the sampled frames (Appendix D.1). Majority voting, a labeling rule of prior video self-play, would assign a wrong label to 52.5% of 7,420 Solver groups (Figure 7A). In 26.3% of all groups a rollout chooses the correct option but loses the vote, as in Figure 1a, and in 14.8% all eight rollouts agree on a wrong option, which agreement filtering would keep. The wrong-label share is 56.8% at 4B and 48.2% at 8B, so consensus labels would mislabel about half of the questions at both scales. Evidence credit shows the same contrast: about half of the rollouts that the content-blind overlap $\max_k \mathrm{IoU}(I_k, e_c^\star)$ would credit earn it only on lines that state another event, and r_{grd} withholds that credit (Figure 7B).

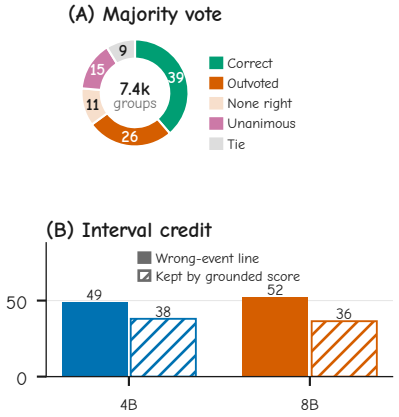


Figure 7: Execution labels. (A) Majority vote versus execution labels on 7,420 Solver groups (%). (B) Rollouts credited only on wrong-event lines, and credit kept by r_{grd} (%).

Training value of evolved environments. Whether the event dynamics that rule selection adds also yield better training data can be tested apart from the training loop, by fine-tuning the 4B base model on the banks of each run (Table 4; Appendix E.2). Both sources improve over the base model in all columns, so execution-labeled data transfer to real videos even without self-play. Data from the evolved rules score higher in every column, so the data produced under rule selection transfer better.

Table 4: Supervised fine-tuning on bank data. The 4B base model fine-tuned for one epoch on the banks of runs without or with rule selection.

Training data	Temporal Grounding	Video QA & Reasoning	Overall
Base model	25.08	54.68	44.81
Initial rules	30.73	56.71	48.05
Evolved rules	**31.78**	**57.09**	**48.65**

5 Conclusion

We presented Tempo-Zero, a self-improvement framework that grounds video self-play in an executable program library, whose programs render synthetic videos together with ground-truth options and intervals, so supervision stays verifiable without external video corpora or majority voting. A single policy acts as both Questioner and Solver and is trained on execution-grounded rewards, while a curriculum allocates tasks by pass rate and the policy selects rule changes. Trained only on simple programs, without external video data or supervised fine-tuning, Tempo-Zero outperforms base models and recent self-play baselines across twelve real-world benchmarks at different model scales, especially in temporal grounding.

ETHICS STATEMENT

All training videos are rendered from synthetic programs, so training uses no human subjects, personal data, or scraped video. Evaluation uses twelve public benchmarks under their released protocols. Appendix G discusses limitations and broader impact.

REPRODUCIBILITY STATEMENT

Algorithm 1 in Appendix A.1 gives the self-play loop, and Table 5 lists the training configuration shared by the 4B, 8B, 30B-A3B, and Gemma models. Appendices A.3 and A.4 describe compute and the evaluation protocol. Appendix B specifies the twelve environment families, frame sampling, and the input and output protocols of both roles, and Appendix C defines the rewards, the question bank, and the optimization. Appendix E defines the ablation variants, and Appendix F lists the prompts together with a Solver response. Code will be made publicly available.

AI USE STATEMENT

We used generative AI tools to draft and edit the paper text, to format references, tables, and figures, and to create and edit software code. The authors reviewed all AI-assisted text and code and take responsibility for the full content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Luke Bailey, Kaiyue Wen, Kefan Dong, Tatsunori Hashimoto, and Tengyu Ma. Scaling self-play with self-guidance, 2026. URL https://arxiv.org/abs/2604.20209.

Ellis Brown, Arijit Ray, Ranjay Krishna, Ross Girshick, Rob Fergus, and Saining Xie. SIMS-V: Simulated instruction-tuning for spatial video understanding, 2025. URL https://arxiv.org/abs/2511.04668.

Yukang Chen, Wei Huang, Baifeng Shi, Qinghao Hu, Hanrong Ye, Ligeng Zhu, Zhijian Liu, Pavlo Molchanov, Jan Kautz, Xiaojuan Qi, Sifei Liu, Hongxu Yin, Yao Lu, and Song Han. Scaling RL to long videos. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2507.07966.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2401.01335.

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. Video-R1: Reinforcing video reasoning in MLLMs. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2503.21776.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24108–24118, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Fu_Video-MME_The_First-Ever_Comprehensive_Evaluation_Benchmark_of_Multi-modal_LLMs_in_CVPR_2025_paper.html.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. TALL: Temporal activity localization via language query. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. URL https://openaccess.thecvf.com/content_ICCV_2017/papers/Gao_TALL_Temporal_Activity_ICCV_2017_paper.pdf.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. VisPlay: Self-evolving vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/papers/He_VisPlay_Self-Evolving_Vision-Language_Models_CVPR_2026_paper.pdf.

Zefeng He, Xiaoye Qu, Yafu Li, Siyuan Huang, Daizong Liu, and Yu Cheng. VideoSSR: Video self-supervised reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26530–26540, 2026b. URL https://arxiv.org/abs/2511.06281.

Kairui Hu, Penghao Wu, Fanyi Pu, Wang Xiao, Xiang Yue, Bo Li, Yuanhan Zhang, and Ziwei Liu. Video-MMMU: Evaluating knowledge acquisition from multidisciplinary professional videos.

In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27798–27828. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.acl-long.1281/.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2508.05004.

De-An Huang, Shijia Liao, Subhashree Radhakrishnan, Hongxu Yin, Pavlo Molchanov, Zhiding Yu, and Jan Kautz. LITA: Language instructed temporal-localization assistant. In *European Conference on Computer Vision*, 2024. URL https://link.springer.com/chapter/10.1007/978-3-031-73039-9_12.

Shiqi Huang, Ziyue Wang, Zhongrong Zuo, Han Qiu, Qi She, and Bihan Wen. EvoVid: Temporal-centric self-evolution for video large language models, 2026b. URL https://arxiv.org/abs/2605.21931.

Alex Iacob, Andrej Jovanović, William F. Shen, Daniel Burkhardt, Meghdad Kurmanji, Nurbek Tastan, Lorenzo Sani, Niccolò Alberto Elia Venanzi, Ambroise Odonnat, Zeyu Cao, Bill Marino, Xinchi Qiu, and Nicholas D. Lane. The Red Queen Gödel Machine: Co-evolving agents and their evaluators. *arXiv preprint arXiv:2606.26294*, 2026.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 2021. URL http://proceedings.mlr.press/v139/jiang21b.html.

Songtao Jiang, Sibo Song, Chenyi Zhou, Yuan Wang, Ruizhe Chen, Tongkun Guan, Ruilin Luo, Yan Zhang, Zhihang Tang, Yuchong Sun, Hang Zhang, Zhibo Yang, Shuai Bai, Junyang Lin, and Zuozhu Liu. Learning transferable temporal primitives for video reasoning via synthetic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Jiang_Learning_Transferable_Temporal_Primitives_for_Video_Reasoning_via_Synthetic_Videos_CVPR_2026_paper.html.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 706–715, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Krishna_Dense-Captioning_Events_in_ICCV_2017_paper.html.

Dingming Li, Yingxiu Zhao, Xinrui Cheng, Kangheng Lin, Hongbo Peng, Hongxing Li, Zixuan Wang, Yuhong Dai, Haodong Li, Jia Wang, Yukang Shi, Liang Zhao, Jianjian Sun, Zheng Ge, Xiangyu Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. SpatialEvo: Self-evolving spatial intelligence via deterministic geometric environments, 2026a. URL https://arxiv.org/abs/2604.14144.

Haodong Li, Tianfei Ren, Xiaoxiao Ma, Chunmei Qing, Zhen Fang, Sipeng He, Ziyu Guo, Haoyu Wu, Juanxi Tian, Yihang Zou, Ruichuan An, Dongzhi Jiang, Boxue Yang, Ji Xie, Xu Huang, Wenhao Yan, Jialv Zou, Zhengrong Yue, Yaxin Luo, Xiaotong Li, Yuzhu Wang, Junyan Ye, Jinjing Zhao, Zehui Chen, Lin Chen, Renye Yan, Feng Zhao, and Pheng-Ann Heng. VideoCoCo: Code-as-CoT for physically-consistent video generation via an agentic dual-engine system, 2026b. URL https://arxiv.org/abs/2607.27380.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: Self-play in corpus environments improves reasoning, 2025a. URL https://arxiv.org/abs/2510.24684.

Bo Liu, Simon Yu, Yiding Jiang, Ao Qu, Andrew Zhao, Zichen Liu, Junsu Kim, Zijian Zhou, Seungone Kim, Tongzheng Ren, Mickel Liu, Hanfei Yu, Zhaorun Chen, Weiyan Shi, Paul Pu Liang, Luke Zettlemoyer, Yejin Choi, and Natasha Jaques. SPADE: Self-play in adaptive synthetic executable environments, 2026a. URL https://arxiv.org/abs/2608.19197.

Wenqi Liu, Yunxiao Wang, Shijie Ma, Meng Liu, Qile Su, Tianke Zhang, Haonan Fan, Changyi Liu, Kaiyu Jiang, Jiankang Chen, Kaiyu Tang, Bin Wen, Fan Yang, Tingting Gao, Han Li, Yinwei Wei, and Xuemeng Song. VideoTemp-o3: Harmonizing temporal grounding and video understanding in agentic thinking-with-videos. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://arxiv.org/abs/2602.07801.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025b. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Jiahao Meng, Xiangtai Li, Haochen Wang, Yue Tan, Tao Zhang, Lingdong Kong, Yunhai Tong, Anran Wang, Zhiyang Teng, Yujing Wang, and Zhuochen Wang. Open-o3-Video: Grounded video reasoning with explicit spatio-temporal evidence. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.20579.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17473–17498. PMLR, 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Tianyuan Qu, Longxiang Tang, Bohao Peng, Senqiao Yang, Bei Yu, and Jiaya Jia. Does your vision-language model get lost in the long video sampling dilemma? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20889–20899, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Qu_Does_Your_Vision-Language_Model_Get_Lost_in_the_Long_Video_ICCV_2025_paper.html.

Hanoona Rasheed, Abdelrahman Shaker, Anqi Tang, Muhammad Maaz, Ming-Hsuan Yang, Salman Khan, and Fahad Shahbaz Khan. VideoMathQA: Benchmarking mathematical reasoning via multimodal understanding in video. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VI4kGUfPio.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Han Wang, Yi Yang, Jingyuan Hu, Minfeng Zhu, and Wei Chen. V-Zero: Self-improving multimodal reasoning with zero annotation. *arXiv preprint arXiv:2601.10094*, 2026a. URL https://arxiv.org/abs/2601.10094.

Hanyang Wang, Yimo Cai, Weiliang Chen, Jiawei Chi, Haowen Sun, Qiyu Dai, Yi-Hsin Hung, Xingzhuo Guo, Jinshan Ren, Runmao Yao, Ziwei Liu, Mingsheng Long, Yueqi Duan, Jun Gao, Jiangran Lyu, Fangfu Liu, and Jialong Wu. Code as worlds: Agentic discovery of executable world representations for physical reasoning, 2026b. URL https://arxiv.org/abs/2608.27549.

Qi Wang, Yanrui Yu, Ye Yuan, Rui Mao, and Tianfei Zhou. VideoRFT: Incentivizing video reasoning capability in MLLMs via reinforced fine-tuning. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.12434.

Qinsi Wang, Bo Liu, Tianyi Zhou, Jing Shi, Yueqian Lin, Yiran Chen, Hai Helen Li, Kun Wan, and Wentian Zhao. Vision-Zero: Scalable VLM self-evolution via multi-agent self-play. In *International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=s00SNXREV6.

Wenyi Wang, Piotr Piękos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel Machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *International Conference on Learning Representations*, 2026d.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. LongVideoBench: A benchmark for long-context interleaved video-language understanding. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/329ad516cf7a6ac306f29882e9c77558-Abstract-Datasets_and_Benchmarks_Track.html.

Junbin Xiao, Angela Yao, Yicong Li, and Tat-Seng Chua. Can I trust your answer? Visually grounded video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2309.01327.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. CLEVRER: CoLlision events for video REpresentation and reasoning. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1910.01442.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2401.10020.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, 2026a.

Ruixu Zhang, Deyi Ji, Lanyun Zhu, Xuanyi Liu, Yuxin Meng, Ruihang Chu, and Yujiu Yang. Video-Zero: Self-evolution video understanding, 2026b. URL https://arxiv.org/abs/2605.14733.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute Zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems*, 2025a. URL https://arxiv.org/abs/2505.03335.

Yilun Zhao, Haowei Zhang, Lujing Xie, Tongyan Hu, Guo Gan, Yitao Long, Zhiyuan Hu, Weiyuan Chen, Chuhan Li, Zhijian Xu, Chengye Wang, Ziyao Shangguan, Zhenwen Liang, Yixin Liu, Chen Zhao, and Arman Cohan. MMVU: Measuring expert-level multi-discipline video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8475–8489, 2025b. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Zhao_MMVU_Measuring_Expert-Level_Multi-Discipline_Video_Understanding_CVPR_2025_paper.pdf.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. MLVU: Benchmarking multi-task long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13691–13701, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zhou_MLVU_Benchmarking_Multi-task_Long_Video_Understanding_CVPR_2025_paper.html.

Yanyan Zou, Chao Chen, Tung Sum Thomas Kwok, Jiayu Yang, Yunyun Hou, Jingmin Zhu, Zehang Luo, Shengjie Li, Yinhong Liu, Yingjia Wan, Chengzu Li, Tao Yu, Zhijiang Guo, Jiachen Liu, Sirui Wang, and Nan Duan. Data-centric recursive improvement for foundation models: A survey. *Preprints*, 2026. doi: 10.20944/preprints202609.0154.v1. URL https://doi.org/10.20944/preprints202609.0154.v1.

A Training Configuration and Evaluation

A.1 Self-Play Algorithm

Algorithm 1 outlines the training loop of the shared vision-language policy. BATCH draws the next batch from the allocation that the curriculum of §3.3 fixes for the round (Eq. 6) and renders it ahead of the step that uses it. The bank stores every measured canonical question and at most one gated proposal per instance, preselected by task rarity. At capacity, eviction first reduces overrepresented tasks and families and then removes the oldest measurements (Appendix C.2). A single AdamW optimizer state persists across both roles and across rounds.

Algorithm 1 Tempo-Zero training loop

Require: policy π_θ, frozen reference policy π_{ref}, program library $\mathcal{P}$, empty bank $\mathcal{B}$

for round $r = 1, \dots, R$ with playback scale λ_r **do**
  fix the allocation $\omega_{f,a}$ of round r — Eq. 6
  for each step t **do**
    $\mathcal{W} \leftarrow$ BATCH: next batch of the round's allocation, screened and rendered
    sample proposals $q \sim \pi^{\text{Q}}_{\theta_t}(\cdot \mid w)$ for each $w \in \mathcal{W}$; compute gate $g(q)$
    $\hat{\mathcal{Q}} \leftarrow$ gated proposals, at most one per instance, preselected by task rarity
    $\mathcal{Q} \leftarrow \{\bar{q}_w\}_{w \in \mathcal{W}} \cup \hat{\mathcal{Q}} \cup$ questions replayed from $\mathcal{B}$
    sample M rollouts $y \sim \pi^{\text{S}}_{\theta_t}(\cdot \mid v, q)$ for each gated proposal and each $q \in \mathcal{Q}$; compute $p(q)$
    compute r^{Q} for all proposals and r^{S} for rollouts on $\mathcal{Q}$ — Eqs. 3–5
    record measured canonical questions and $\hat{\mathcal{Q}}$ in $\mathcal{B}$; evict overrepresented tasks and families, then the oldest
    mean-center rewards within each group; omit groups whose rewards are all equal
    $\theta_{t+1} \leftarrow$ one GRPO update on Solver groups over $\mathcal{Q}$ and Questioner groups per instance
  end for
  if $r < R$ **then**
    move each pair's level by its decayed pass rate in $\mathcal{B}$
    for each family, the policy sets task weights $w_{f,a}$ and keeps u_f or selects a rule change from its menu
    the executor validates and previews each change; an invalid change keeps the previous program
  end if
end for
return π_θ

A.2 Training Configuration

Table 5 lists the settings shared by the 4B, 8B, 30B-A3B, and Gemma configurations. The Questioner and Solver contribute separate reward groups to one combined update of the LoRA adapters on the language model, and rule selection after each round runs at inference only. The instruction-tuned checkpoint, obtained by disabling the adapters, serves as the reference policy for KL regularization.

The scheduled budget comprises 3,200 instances and at most 12,800 Questioner proposals. Every canonical question, gated proposal, and replayed question is answered eight times ($M = 8$) under the policy snapshot that generated the proposals. These responses give every gated proposal its pass rate for the Questioner reward, and the responses to canonical questions, preselected proposals, and replayed questions also train the Solver. Groups whose rewards do not differ supply pass rates but no gradient, and both roles are scored before the shared parameters change.

A.3 Compute Resources

Each training run uses eight GPUs. A single persistent inference engine serves rollouts for both roles, while video rendering runs asynchronously in a CPU worker pool scheduled one step ahead of the policy update.

A.4 Evaluation Protocol

Benchmark suites. Video reasoning is evaluated on LongVideoReason (Chen et al., 2025), MMVU (Zhao et al., 2025b), VideoMathQA (Rasheed et al., 2026), and VideoMMMU (Hu et al.,

Table 5: Training configuration. Settings shared by the 4B, 8B, 30B-A3B, and Gemma configurations.

Component	Setting
Backbone	Qwen3-VL-4B/8B/30B-A3B-Instruct and Gemma-4-E4B-it; LoRA with rank 64 and $\alpha = 128$ on the q/k/v/o projections of the language model; the vision tower and the A3B experts carry no adapter
Environments	12 environment families, 3 difficulty levels; 5 rounds of 20 steps
Batch	32 instances per step; playback scale $\lambda_r = 1.0, 1.25, 1.5, 1.75, 2.0$ for rounds $r = 1, \ldots, 5$
Questioner	4 proposals per instance; one to three located evidence lines
Solver	$M = 8$ rollouts per question; one to eight evidence lines; criteria are the target plus the criteria of the events its answer rests on; at most 72 groups per step (32 canonical, up to 32 preselected proposals, 8 replayed)
Question bank	Capacity 640, shared across rounds; up to 8 replayed questions per step; at most one preselected proposal per instance; one prior example in the Questioner prompt
Option count	Cycles across (6, 6, 6, 5, 4)
Curriculum	Coverage share $\kappa = 0.30$; pass-rate decay 0.90 per step; level thresholds $\rho_{\mathrm{adv}} = 0.80$ and $\rho_{\mathrm{ret}} = 0.20$ on the decayed pass rate, applied with at least 3 questions; task weights start equal and change by at most a factor of 2 per round
Rule menu	At most 3 alternative configurations besides keep; one selection per family per round
Optimizer	AdamW, learning rate 5×10^{-5}, weight decay 10^{-2}
Schedule	Constant learning rate without warmup; single PPO epoch per step; one combined update over both roles, 100 updates per run
Regularization	Low-variance KL with coefficient $\beta_{\mathrm{KL}} = 10^{-3}$ against the base model with adapters disabled; clipping threshold 0.2; gradient norm cap 0.5; advantages mean-centered without standard-deviation normalization
Precision	fp32 master weights; bf16 compute and rollout
Sampling	Temperature 1.0; top-p 0.99; unrestricted top-k
Token limits	16,384 prompt tokens; 2,048 response tokens; context ceiling 24,576
Visual input	Uniform sampling at 2 fps, 4 to 64 frames, so clips longer than 32 s fall below 2 fps; 3,136 to 100,352 pixels per frame
Checkpointing	Saved after every round (every 20 steps)
Holdout	128 synthetic instances, evaluated before training and after every round; used for neither training nor curriculum
Hardware	8 GPUs per run

2026), scored by question-answering accuracy (%). Temporal grounding is evaluated on ANet-RTL (Huang et al., 2024), ActivityNet Captions (Krishna et al., 2017), Charades-STA (Gao et al., 2017), and NExT-GQA (Xiao et al., 2024). The first three report mean Intersection over Union (mIoU) between predicted and ground-truth intervals together with recall at IoU thresholds 0.3, 0.5, and 0.7, and NExT-GQA reports grounded accuracy at IoU 0.3 and 0.5, which requires both a correct answer and temporal overlap. Long-video understanding is evaluated on LongVideoBench (Wu et al., 2024), MLVU (Zhou et al., 2025), Video-MME-Long (Fu et al., 2025), and LSDBench (Qu et al., 2025), which probe visual context over extended durations and are scored by question-answering accuracy (%).

Final evaluation. Final numbers use the checkpoint from the fifth training round on the complete test splits of all twelve benchmarks. All models, including the baselines and ablation variants, are evaluated under the same protocol, which follows Zhang et al. (2026b) and the official metrics of each benchmark.

B Executable Environments and Interfaces

B.1 Environment Coverage and Design

Table 7 summarizes the program library, and Table 6 lists the twelve tasks. Each environment family implements a deterministic state-transition system, rendering routines, query templates, and analytical verification routines. Difficulty levels govern the number of interacting entities and the operation complexity, while the playback scale sets the temporal duration. Levels enter in rotation and thereafter follow the curriculum, and option counts cycle over a fixed sequence. Figure 8 shows state transitions from three representative families.

Table 6: Twelve tasks and answer types. Each task is followed by its answer type, shown in gray.

Task	Type	Task	Type	Task	Type	Task	Type
Event grounding	interval	Event order	sequence	Event count	count	Extremum	entity
State interval	interval	Path sequence	sequence	State change	count	Occluded identity	entity
Adjacent event	event	Interval relation	relation	Rate trend	comparison	Motion direction	direction

Table 7: Twelve environment families. Each family supports three difficulty levels; the columns summarize its observable dynamics and the reasoning targets it supports.

Family	Observable dynamics	Reasoning targets
Billiards	Kinematic motion, elastic contact, and velocity shifts	Trajectory direction, event order, state transitions
Block stacking	Sequential placement and physical support	Object counts, stacking order, spatial relations
Checkout queue	Arrival, service progression, and departure	Queue duration, entity counts, interval relations
Dominoes	Cascading propagation along contact dependencies	Chain continuation, trigger ordering, collapse speed
Ferris wheel	Coupled rotating structures and cyclic cabins	Relative angular rates, phase shifts, event timing
Filling glasses	Liquid level shifts and volumetric transfers	Continuous quantities, extrema, volume change intervals
Maze run	Discrete path progression and visited markers	Route reconstruction, intersection counts, milestone order
Parking lot	Vehicle ingress, stall occupancy, and egress	Occupancy states, dwell durations, interval intersections
Shell game	Continuous occlusion, permutations, and transfers	Object tracking, identity recovery, hidden states
Tag spread	Contact-induced state propagation across agents	Infection ordering, transmission delays, final states
Traffic lights	Periodic signal phases and permitted trajectories	Phase duration counts, rule compliance, interval localization
Tunnel trains	Motion across visible and occluded segments	Transit speed, occluded intervals, exit order

B.2 Video Frame Sampling and Temporal Coordinates

Frames are sampled uniformly over the video at a fixed target rate up to a fixed maximum count, always including the initial and final states. A criterion's target must be visible in at least three sampled frames. Every ground-truth value is recomputed from the sampled frames, and a criterion is discarded when its value cannot be recomputed from those frames or disagrees with the simulation. The static-question screen records three outcomes: changed under an evaluable transformation, unchanged under both evaluable transformations, or undetermined. The second outcome is excluded, and undetermined criteria remain subject to the normal visibility and validity checks. A transformation cannot be evaluated when reversal creates tied events or leaves the query's supported domain.

All ground-truth temporal labels use the effective time coordinate exposed to the model. For F selected frames spanning the rendered video duration d, which playback scale λ_r stretches, the observation manifest defines

$$f_{\text{label}} = \frac{F-1}{d}, \qquad t_i = \frac{i}{f_{\text{label}}}, \quad i = 0, \ldots, F-1. \tag{7}$$

The input processor consumes this effective frame rate and the selected frames directly, without a second temporal downsampling, so verification intervals map to the same discrete timestamps that the model sees.

B.3 Verification Checks on the Training Environments

To measure how often the two checks of §3.1 apply, we read the check records of the 5,074 distinct instances that the three Qwen3-VL training runs draw from the twelve families, 2,599 at level 0, 1,179 at level 1, and 1,296 at level 2. A level-2 video holds 22 key events over 119 seconds on average, against 12 over 36 seconds at level 0 (Figure 9A). Frames are selected uniformly over each video,

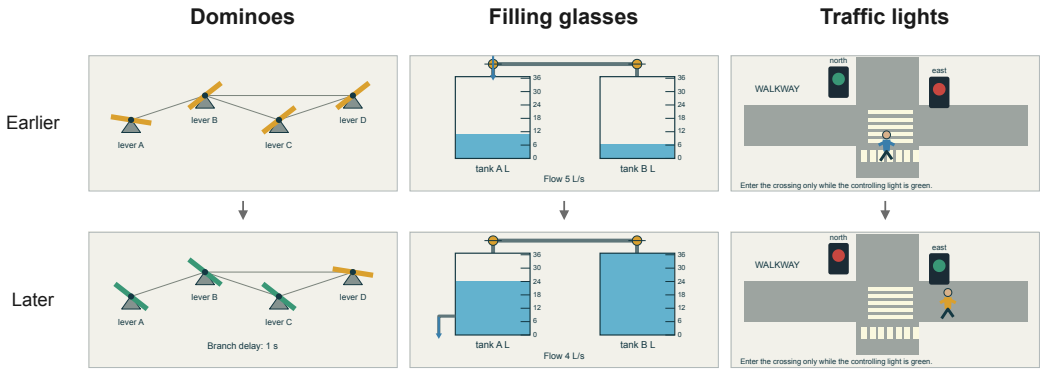


Figure 8: Executable environments. Rendered frame pairs from Dominoes, Filling glasses, and Traffic lights; the labels inside the frames are part of the rendered video.

independently of where events occur, at no more than 2 fps and 64 frames per video. A level-0 video thus has 58.1 frames on average at an effective rate of 1.69 fps, and a level-2 video 64.0 frames at 0.62 fps (Eq. 7). Table 8 shows that the share of recorded events that start and end between two sampled frames grows from 2.82% at level 0 to 28.56% at level 2. A label read from the simulation clock would therefore often describe a state that no sampled frame shows.

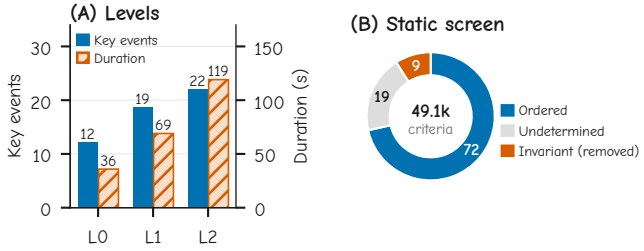


Figure 9: Training environments. (A) Key events and duration by level. (B) Static-question screen classes in %; only invariant criteria are removed.

Two parts of the environment prevent this. First, the key events that questions refer to are limited to events that at least one sampled frame shows and that lie at least two sampled frames apart, and an instance must keep at least 3, 5, or 7 such events at levels 0, 1, and 2. Every one of the 82,392 key events in these instances therefore appears on a sampled frame. Second, the recomputation discards 403 of the 49,525 compiled criteria: in 208 the recomputed value disagrees with the simulation, and 195 cannot be recomputed from the frames. Disagreements concentrate in Event count and Extremum, with 110 and 68 criteria, and the most frequent case counts three events on the simulation clock, of which the sampled frames show two. The discarded share grows from 0.19% at level 0 to 2.13% at level 2, as the share of events between frames does.

The static-question screen removes only 9.01% of the remaining 49,122 criteria, those whose answer stays the same under both reversal and permutation (Figure 9B). Nearly all criteria of Event grounding, Event order, and State interval pass the screen, with at most 0.13% removed. The removed criteria come mainly from Extremum and Event count, where 35.69% and 34.04% of each task's criteria keep the same answer under both transformations, since many counts and comparisons do not depend on event order.

B.4 Role Inputs and Output Protocols

The Questioner receives the rendered frames, the execution metadata with observation timestamps, the askable criteria, same-task questions from the bank, and the Solver's pass rate per task; its reward contains no localization term. Each criterion specifies a target entity and the relation evaluated on it, a question stem, the ground-truth option, the ground-truth interval, and a distractor domain with a resolution floor. The Solver receives only the frames, the question text, and the options.

When proposing, the Questioner names one verified criterion, writes the stem and the options, and cites the events the criterion depends on as located evidence lines before its answer payload. The gate

Table 8: Verification checks on the training environments. Instances per level, mean sampled frames and effective frame rate, share in % of recorded events that fall between two sampled frames, criteria compiled and discarded by the recomputation, and shares in % of the static-question screen classes among the criteria that pass the recomputation. Invariant criteria are removed, and ordered and undetermined criteria are kept.

		Sampled frames		Events	Recomputation			Static-question screen		
Level	**Instances**	Count	fps	**between**	Criteria	Disagree	Unverified	Ordered	Invariant	Undet.
0	2,599	58.1	1.69	2.82	25,338	23	25	67.93	10.76	21.31
1	1,179	63.8	1.00	10.02	11,623	44	44	74.33	7.29	18.38
2	1,296	64.0	0.62	28.56	12,564	141	126	76.21	7.02	16.77
Overall	5,074	60.9	1.26	16.60	49,525	208	195	71.51	9.01	19.49

accepts a proposal only if exactly one option equals the ground-truth option $y^\star$ and all distractors lie in the criterion's domain, separated at the declared temporal or numerical resolution. The stem must also match the task specification, name the criterion's subjects, keep its qualifiers, state no ground-truth value, and end in a single question mark. The environment runtime then determines the final option permutation and the answer letter. Criteria without enough distinct distractors are excluded from the askable set, and the Questioner's located lines are never shown to the Solver.

The Solver is prompted to write up to eight evidence lines before a single option character inside `<answer>`. Each evidence line starts with a JSON list of `[start, end]` pairs in decimal seconds, followed by an observed state transition; a recurring event places all of its intervals in that one list, and the final line states the resolving condition. Static in-context demonstrations define these output structures (Appendix F).

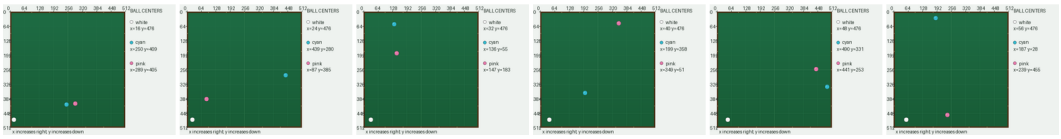
B.5 Rule Selection and Execution Determinism

After each round except the last, the environment runtime composes for every family a menu of at most three alternative rule configurations besides the current one. Each entry moves one rule to a value the family has not run, and from the third round on, entries that move a pair of rules lead the menu. The policy receives the menu, the rule meanings, the current executable graph, the editable schema, prior changes, and measurements under the current configuration and playback scale. Canonical and Questioner measurements remain separate, and recorded responses and evidence scores are provided as feedback. The output selects a menu entry and sets the task weights $w_{f,a}$ of the family; Shell game additionally supports operation-list, dependency, and operation-parameter edits of its operation graph. Each weight changes by at most a factor of two per round, and only for tasks with at least three current-configuration questions. The interface permits neither arbitrary program rewriting nor independent changes to ground truth. The executor checks field validity, parameter domains, references, finite expansion, and acyclicity, and a joint preview executes the proposed configuration and graph together before rendering and recompiling the resulting supervision. Invalid or failed outputs keep the previous program. Task levels move by the pass-rate thresholds of §3.3 and task weights by the policy's bounded selection.

Selected configurations and program graphs determine future traces, renderings, criteria, and answers. Instance records preserve the original generation context: replay resolves the saved instance identifier through its frame manifest and keeps the question's original scoring context, so later environment edits do not alter earlier questions. The curriculum constants, namely the pass-rate decay, the coverage share κ, the level thresholds and their minimum question count, and the weight bound, are fixed across runs (Table 5).

Example of a rule change. Figure 10 shows one rule change in Billiards with the seed and level held fixed. Changing the side cushion from specular to retroreflective reflection sends the pink ball back along its incoming path. The canonical question, "Which table quadrants does the pink ball enter on its first 12 quadrant changes, in order?", stays the same, but re-execution changes its ground-truth option from "lower-left, upper-left, upper-right, lower-right" repeated three times to "lower-left, lower-right, upper-right, lower-right" repeated three times, together with the 12 ground-truth intervals. The sampled frames coincide up to 4.0 s and differ from 7.9 s on, so the new rule yields new event dynamics whose labels remain exact.

Before: specular side cushions



After: retroreflective side cushions

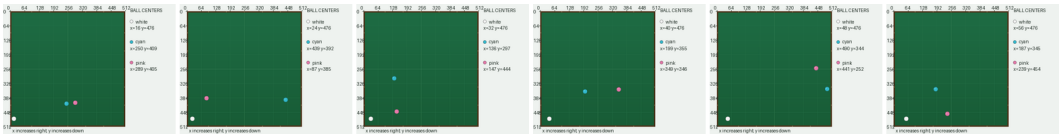


Figure 10: A rule change in Billiards. Frames at 0.0, 4.0, 7.9, 11.9, 15.9, and 19.8 s of the same instance before and after the rule change; the overlays are part of the rendered frames.

C Rewards, Question Bank, and Optimization

C.1 Reward Terms

This section details the reward terms of Eq. 4.

Learnability. Let s of the M Solver rollouts on question q be correct. With pass rate $p = s/M$ as the estimated probability that one rollout is correct, the Bernoulli variance $p(1-p)$ peaks at $p = 1/2$ and vanishes when all rollouts are correct or all are wrong. Laplace smoothing replaces p by $\tilde{p} = (s+1)/(M+2)$, and the learnability reward

$$r_{\text{learn}}(q) = 4\,\tilde{p}\,(1-\tilde{p}) = \frac{4(s+1)(M-s+1)}{(M+2)^2} \tag{8}$$

equals one at $s = M/2$ and stays positive at $s \in \{0, M\}$. With $M = 8$, it takes the values 0.36, 0.64, 0.84, 0.96, and 1 for $s = 0, 1, 2, 3, 4$ and is symmetric around $s = 4$.

Evidence matching. The Solver's evidence lines are parsed with the grammar of Appendix B.4. Each line is assigned to at most one criterion of $\mathcal{C}_q$ and each criterion to at most one line, so that the assignment is one-to-one; $k(c)$ denotes the line assigned to criterion c. The indicator δ_c equals one when the assigned line states the event of c and zero otherwise; a criterion without an assigned line has $\delta_c = 0$. A line states the event of c when it asserts the recorded value of c without negation, in a clause that names a subject of c and a term of its relation. An ordered value must list its items in the recorded order, and a duration must be stated as an interval. With $m = \sum_{c \in \mathcal{C}_q} \delta_c$ matched criteria, precision $P = m/L$ and recall $R = m/|\mathcal{C}_q|$ give

$$r_{\text{evid}} = \frac{2PR}{P+R} = \frac{2m}{L + |\mathcal{C}_q|}, \tag{9}$$

which is zero when no criterion is matched. The grounded score averages $\delta_c \,\mathrm{IoU}(I_{k(c)}, e_c^\star)$ over $\mathcal{C}_q$, where the IoU of two finite unions of intervals is the length of their intersection divided by the length of their union.

Novelty. The novelty reward discourages the Questioner from repeating one question pattern. Every proposal q has a task signature

$$\sigma(q) = \big(f,\ a,\ \ell,\ o,\ N_e\big), \tag{10}$$

where f is the environment family, a the task, ℓ the difficulty level, o the transition pattern, and N_e the number of interacting entities. With $N_{\mathcal{B}}(\sigma)$ questions of signature σ in the bank,

$$r_{\text{nov}}(q) = \exp\!\big(-N_{\mathcal{B}}(\sigma(q))/c_{\text{nov}}\big), \qquad c_{\text{nov}} = 10, \tag{11}$$

so an unseen signature receives $r_{\text{nov}} = 1$ and the reward decays as questions with the same signature accumulate. The counts are kept in a table over discrete signatures and require no learned embedding.

C.2 Question Bank

Every measured canonical question enters the bank, so each instance contributes at least one stored question. Among the proposals that pass gate $g(q)$ (§3.2), the bank preselects at most one per instance, favoring the task that is rarest in the bank, and stores it once measured; proposals with $g(q) = 0$

are neither answered nor stored. Each entry keeps the question, its verification context, the rule configuration and program graph of its instance, and the instance identifier. Replay reloads the rendered frames of the stored instance and answers them with the current policy. For every (family, task) pair and level, the bank keeps exponentially decayed pass-rate statistics, including counts of solved ($p = 1$), learnable ($0 < p < 1$), and unsolved ($p = 0$) questions, which the curriculum of §3.3 and the rule-selection prompt read. The bank holds 640 questions shared across rounds; at capacity, eviction first removes questions of overrepresented tasks and families and then the oldest measurements, which keeps all twelve tasks represented.

C.3 Group Relative Normalization and Policy Optimization

Policy parameters θ are shared between Questioner π_θ^{Q} and Solver π_θ^{S} and updated with Group Relative Policy Optimization (GRPO). The Solver forms groups $\mathcal{G}_{\mathrm{S}}$ of $M = 8$ rollouts per question, and the Questioner forms groups $\mathcal{G}_{\mathrm{Q}}$ of 4 proposals per instance. The advantage of each sample is its reward minus the group mean,

$$\hat{A}_i = r_i - \frac{1}{|\mathcal{G}|} \sum_{j \in \mathcal{G}} r_j, \tag{12}$$

without division by the group standard deviation, which keeps the scale of reward differences (Liu et al., 2025b). Groups whose rewards are all equal have zero advantage and are omitted from backpropagation. Both roles use the clipped surrogate objective with a Kullback–Leibler penalty $\beta_{\mathrm{KL}} = 10^{-3}$ against the frozen reference policy π_{ref}, the base model with adapters disabled. Training performs one combined Questioner–Solver update per step with AdamW at learning rate 5×10^{-5}. The shared policy is held fixed throughout proposal sampling, Solver sampling, and reward evaluation, so each Questioner reward measures the current Solver without an intermediate parameter change. Each role's advantages are centered within its own groups, and the optimizer receives the eligible groups of both roles.

D Analysis Details

The analyses of §4.4 use the recorded proposals, rollouts, rule decisions, and held-out evaluations of the 4B and 8B training runs.

D.1 Majority Voting and Interval Credit by Backbone

For every Solver group, majority voting takes the option chosen most often among the eight rollouts, and a tie leaves no majority label. Table 9 compares this label with the execution label. Every one of these groups is scored against a criterion that passed the recomputation and the static-question screen of §3.1, and its ground-truth value equals the value recomputed on the sampled frames. Re-executing the programs of all instances reproduces the same traces, frame selections, and discarded criteria, so the execution labels are reproducible. The majority label is wrong for 56.8% and 48.2% of the groups at 4B and 8B and for 50.8% to 53.5% at each difficulty level, so it would mislabel about half of the training questions regardless of scale or video length. Wrong labels on which all eight rollouts agree are most frequent at 4B, where agreement filtering would keep a wrong label for 21.3% of the groups. The right half of the table covers all Solver rollouts. More than 90% of them receive some content-blind overlap. In 48.8% and 52.2% of these rollouts at 4B and 8B, the overlap comes only from lines whose statement does not match the criterion, so r_{grd} gives them no interval credit. The grounded score keeps 36% to 38% of the content-blind credit, and the credit it keeps falls only on lines that state the correct event (Eq. 3).

D.2 Held-Out Instances

Each run builds its held-out set once from program seeds that no training round reaches. The 128 instances are balanced over families and levels at playback scales 1.0 and 2.0, and a candidate that fails the generation checks is replaced by another of the same family and level, so model answers never determine membership. After every round, the policy answers the canonical question of each instance eight times. An answer counts as correct when it selects the ground-truth option. Evidence and grounded scores are r_{evid} and r_{grd} of Eq. 3. Table 10 shows that all three metrics exceed the base model after every round at both backbones.

Table 9: Majority voting and interval credit. Left: Solver groups, with shares in % of groups whose majority label is wrong, whose majority label is wrong although at least one rollout chooses the correct option, whose eight rollouts agree on a wrong option, and whose vote ties. Right: Solver rollouts, with the share in % of rollouts credited by the content-blind overlap whose credit falls only on lines with a non-matching statement, and the share in % of the content-blind credit that r_{grd} keeps.

	Majority voting					Interval credit	
Backbone	Groups	Wrong	Outvoted	Unanimous	Tie	Wrong line	Kept
4B	3,703	56.82	23.17	21.33	7.24	48.78	38.01
8B	3,717	48.21	29.32	8.34	10.36	52.21	36.44
Pooled	7,420	52.51	26.25	14.82	8.80	50.50	37.18

Table 10: Held-out instances. Accuracy, evidence score, and grounded score in % on the 128 held-out instances with eight answers each, for the base model (B) and after each of the five rounds; bold marks the best round.

Backbone	**Metric**	B	1	2	3	4	5
4B	Accuracy	22.9	27.9	32.7	32.4	**33.2**	31.1
	Evidence	24.3	40.4	42.1	41.7	43.7	**47.0**
	Grounded	3.1	7.6	8.9	7.1	10.0	**10.8**
8B	Accuracy	25.9	31.3	36.1	40.5	37.5	**42.8**
	Evidence	27.2	38.3	40.3	40.9	40.7	**42.6**
	Grounded	3.4	7.0	10.0	11.3	10.8	**11.6**

D.3 Rule Decisions

Table 11 lists the 96 family decisions of the two runs. A decision keeps the rules either because the policy selects the current configuration or because its output fails a check of Appendix B.5. Every alternative on the menus, 270 in total, compiles into usable instances in the runtime's preview.

Table 11: Rule decisions. Families whose rules change for each round, decisions that keep the rules by the policy's selection or after a failed check, and menu alternatives with usable previews. Each backbone makes 12 decisions per round.

	Changed					Kept		
Backbone	2	3	4	5	Total	Selected	Failed check	**Usable alternatives**
4B	10	12	7	3	32	12	4	129 / 129
8B	9	7	3	5	24	23	1	141 / 141
Total	19	19	10	8	56	35	5	270 / 270

E Controlled Comparisons

E.1 Ablation Variants

Table 12 defines the ablation variants of Table 3. Every variant starts from the same initialization and uses the same training budget, curriculum constants, and evaluation protocol as the full method, and it differs from the full method in exactly one component.

E.2 Training Value of Evolved Environments

The comparison of Table 4 in §4.4 fine-tunes the base Qwen3-VL-4B-Instruct checkpoint for one epoch on the questions stored in the bank at every round of one run, paired with the Solver responses recorded for them. The two runs are the ablation that keeps the initial rules (w/o rule selection) and the full method, whose rules evolve through rule selection; they use the same curriculum and budget and differ only in rule selection. The fine-tuned model does not select rules in this comparison, so differences between rows come from the data that rule selection produced.

Table 12: Ablation variants. Each variant changes one component of the full method.

Variant	Change from the full method	Component tested
w/o novelty reward	Remove the novelty reward r_{nov} from r^{Q}	Diversity of proposed questions
w/o learnability reward	Remove the learnability reward r_{learn} from r^{Q}	Targeting of questions at the Solver's capability
w/o Questioner	Train the Solver only on canonical questions, without Questioner proposals	Self-proposed questions
w/o evidence score	Remove the evidence score r_{evid} from r^{S}	Statement-level F_1 of evidence lines
w/o grounded score	Remove the grounded score r_{grd} from r^{S}	Interval credit on matched lines
Answer reward only	Keep only the answer reward r_{ans} in r^{S}	Both evidence terms
w/o rule selection	Keep the initial rules in every round; the curriculum still reallocates instances	Rule changes

F Prompt Templates and Qualitative Rollout

The Questioner prompt (Listing 1) provides the video frames, the execution metadata, the askable criteria, bank examples of the same task, and the Solver's pass rate per task; the listing shows its task statement and output format. The Solver prompt (Listing 2) specifies the format of evidence lines and answers. Both prompts remain fixed across all experiments; bracketed markers show where task specifications and environment inputs are inserted.

Listing 1: Questioner prompt (task statement and output format).

```
Your mission is to analyze a Video, Program Metadata, Available Task
Criteria, and Existing Questions from the same Source Form to generate one
new, highly discriminative, evidence-grounded multiple-choice question.

The question must expose a meaningful capability gap that is not already
captured by the existing questions. It must be answerable from the video and
verifiable against the program metadata.

[Input data, core principles, and two demonstrations are inserted here.]

### Output Format
Write the located evidence lines first, with no tag around them and no text
before them: one line per event the fact you cite rests on, one to three lines
in all, each opening with the seconds you read that event at in the video. Then
give the proposal between the <answer> and </answer> tags, and write nothing
after </answer>.
[[X, Y]] one located claim
<answer>criterion: cK
question: <one question ending in ?>
options:
A. ...
B. ...</answer>
```

Listing 2: Solver prompt (head and evidence-line rules).

```
Your task is to answer the multiple-choice question from the video and cite the visible
     evidence that determines the answer.

Use only the current video, question, and options. Provide your thinking process as
    located evidence lines, and then give your final answer between the <answer> and
    </answer> tags:
[[X, Y]] one concrete visible claim
[[U, V], [W, Z]] one claim supported by separated intervals
<answer>LETTER</answer>

Requirements:
- Write 1-8 evidence lines before <answer>; do not add unlocated reasoning lines.
- Start every evidence line with its time intervals as a JSON list of
  [start, end] pairs in decimal seconds, for example [[12.30, 25.70]]; put
  separated occurrences of the same event in that one list in time order.
[Remaining rules, worked examples, the question, its options, and the video are
    inserted here.]
```

Representative rollout example. Listing 3 shows a Solver response on a Billiards question. The question asks how the target ball's quadrant changes between the start and the end of the video, with six options, where option `F` is a shift from the upper-left to the lower-left quadrant. The Solver cites two intervals corresponding to the initial and terminal observation frames, and its final line states the conclusion in the form the criterion is checked against.

Listing 3: A Solver response on a Billiards question.

```
[[0.10, 0.60]] The pink ball is positioned in the upper-left quadrant at the start.
[[47.70, 48.00]] By the end, the pink ball is located in the lower-left quadrant,
    showing a movement from upper-left to lower-left.
<answer>F</answer>
```

G Limitations and Broader Impact

Limitations. Three constraints bound this study. First, the instruction-tuned backbones inherit visual and linguistic abilities from pretraining, and the executable environments shape only the self-play stage that follows. Second, the environments cover twelve hand-designed families of physical behavior with fixed task schemas and rule menus, and natural video has greater perceptual variability and commonsense breadth. Third, supervision relies on deterministic rendering and exact temporal coordinates, and the evidence terms credit only statements that match the verification criteria.

Broader impact. Tempo-Zero derives its training signal from executed programs, which reduces the need for scraped video and human temporal annotation. Because every training video is rendered from synthetic programs, self-play uses no private, copyrighted, or personal video data. As in other synthetic training methods, rule selection could be used to steer models toward narrow visual priors. Rule configurations are versioned, so the curriculum of every run can be inspected.