

Self-Evolving Vision Language Models for Structured Visual Understanding From Zero Data

Zongxia Li^{1*} Xiaomin Wu¹ Chongsong Huang^{2*} Hongyang Du^{3*} Xiyang Wu¹

Lantao Yu⁴ Yicheng He⁵ Zhichao Liu¹ Jiarui Zhang⁶ Fuxiao Liu⁷ Jing Xie^{1†}

¹University of Maryland ²Washington University in St. Louis ³Brown University

⁴Adobe ⁵University of Illinois Urbana-Champaign ⁶University of Southern California ⁷NVIDIA

zli12321@terpmail.umd.edu fred.xiaomin.wu@gmail.com chongsong@wustl.edu hongyang_du@brown.edu wuxiyang@umd.edu
lantaoy@adobe.com yh84@uiuc.edu zcliu@terpmail.umd.edu jzhang37@usc.edu fuxiao1@nvidia.com jingxie@terpmail.umd.edu

* Core contribution [†] Corresponding author

Abstract

Self-evolving has emerged as a key paradigm for improving foundational models such as Large Language Models (LLMs) and Vision Language Models (VLMs) with minimal human intervention. While recent approaches have demonstrated that LLM agents can self-evolve from scratch with little to no data, VLMs introduce an additional visual modality that typically requires at least some seed data, such as images, to bootstrap the self-evolution process. In this work, we present Multi-model Multimodal Zero (MM-Zero), the first RL-based framework to achieve zero-data self-evolution for VLM reasoning. Moving beyond prior dual-role (Proposer and Solver) setups, MM-Zero introduces a **multi-role** self-evolving training framework comprising three specialized roles: a **Proposer** that generates structured visual concepts and formulates questions; a **Coder** that translates these concepts into executable code (e.g., Python, SVG) to render visual images; and a **Solver** that performs multimodal reasoning over the generated visual content. All three roles are initialized from the same base model and trained using Group Relative Policy Optimization (GRPO), with carefully designed reward mechanisms that integrate execution feedback, visual verification, and difficulty balancing. Our experiments show that MM-Zero improves average VLM reasoning performance across a wide range of multimodal benchmarks. MM-Zero establishes a scalable path toward self-evolving multi-model systems for multimodal models, extending the frontier of self-improvement beyond the conventional two-model paradigm.

1 Introduction

Self-evolving mechanisms (Xiang et al.; Schmidhuber, 2007; Clune, 2019; Huang et al., 2025a; Yue et al., 2026a) represent a promising frontier for advancing foundational models. Rather than relying on static, human-curated supervision, self-evolving systems learn by independently generating, refining, and learning from their own experiences such as through self-play (He et al., 2025; Wang et al., 2025c) or synthetic data generation (Silver et al., 2016). This paradigm has seen rapid progress in the domain of Large Language Models (LLMs): recent works have shown that LLMs can autonomously improve their reasoning by generating their own training tasks and refining themselves through reinforcement learning (Huang et al., 2025a; Liu et al., 2025) or code execution feedback (Zhao et al., 2025a), achieving strong downstream performance without any human-annotated data.

A natural question is whether similar self-evolving paradigms can be extended to Vision Language Models (VLMs). Post-training of VLMs has traditionally relied on large volumes of expert-curated tasks and labels (Liu et al., 2023b; Li et al., 2025a; Huang et al., 2025b; Li et al., 2025a; Feng et al., 2025; Liu et al., 2023a; Fei et al., 2024), making it costly, labor-intensive, and fundamentally bottlenecked by the availability of human annotation (Yang et al., 2024; Silver et al., 2016). However, extending self-evolution to VLMs is non-trivial. Unlike LLMs, which operate over text alone, VLMs require visual inputs, meaning any self-evolving loop must also generate or supply images, not just questions and answers.

Existing VLM self-evolution approaches (He et al., 2025; Thawakar et al., 2025; Wang et al.,

2026) attempt to address this by adapting the *proposer-solver* pipelines originally designed for LLMs (Huang et al., 2025a; Yue et al., 2026b). In these frameworks, a Proposer proposes visual queries and a Solver answers them. While this eliminates the need for human-annotated labels, the entire iterative process remains rigidly conditioned on pre-existing, collected static image datasets. This merely shifts the bottleneck: the models evolution remains bounded by the distribution, quality, and diversity of the collected image dataset, as well as the human cost required to curate and filter it (He et al., 2025; Thawakar et al., 2025; Wang et al., 2026). Self-evolving VLMs from static images thus introduces an additional burden of sourcing image data, a filtering and collection process that is itself time-consuming and costly (Liu et al., 2023b; Li et al., 2025a, 2026a; Yue et al., 2024b). In contrast, synthetic visual data offers a promising direction for self-evolving VLMs: programmatically rendered scenes can represent structured visual concepts and simulate diverse, controllable scenarios beyond the coverage of curated datasets (Long et al., 2024; Tobin et al., 2017). This principle is well established in robotics, where controlled simulation environments are widely used to train autonomous driving systems on rare or dangerous scenarios that would be costly, unsafe, or impractical to collect in the real world (Dosovitskiy et al., 2017).

We introduce **MM-Zero**, the first **zero-data** self-evolving reinforcement learning framework that trains VLMs without relying on any external data. Instead of conditioning on existing images, questions, or human labels, MM-Zero enables the model to generate its own visual content, starting from simple scenes and progressively creating more complex ones. To achieve this, we expand the traditional dual-role setup into a **multi-role** interaction, with all roles initialized from the same base model and optimized using Group Relative Policy Optimization (GRPO) (Shao et al., 2024): A **Proposer** (Abstract Conception) that formulates diverse topics, high-level visual descriptions, and associated questions; A **Coder** (Visual Synthesis) that translates these concepts into executable code (e.g., Python, SVG) to render structured visual images; A **Solver** (Multimodal Reasoning) that engages with the synthesized visual content to solve complex reasoning tasks.

This multi-role architecture establishes a closed-loop system where abstract language, executable

code, and visual reasoning mutually reinforce one another. Unlike previous approaches where images are fixed, the proposer is encouraged to create *Goldilocks* tasks (Johnston, 1990) that are challenging yet solvable for the coder to render images and the solver to solve. Our Coder receives direct *execution feedback* and visual quality feedback from the solver as rewards. Our solver uses the training data generated by the proposer and coder to perform test-time reinforcement learning (TTRL) (Zuo et al., 2025) to improve itself. To the best of our knowledge, MM-Zero is the first framework to explore self-evolving multimodal reasoning through interactions among more than two roles, requiring zero human-annotated data.

We apply MM-Zero to train Qwen3-VL-Instruct 4B, 8B (Bai et al., 2025) as well as MIMO-VL-7B-SFT (Xiaomi, 2025), demonstrating that visual reasoning capabilities can emerge purely from self-play. Without any real-world training data, MM-Zero achieves average performance gains across visual chart understanding, visual math, and real-world natural-image benchmarks, although some individual benchmarks remain flat or fluctuate. Our main contributions are:

- We propose **MM-Zero**, the first zero-data self-evolving framework for improving VLM reasoning through multi-model self-evolution, requiring no external data.
- We introduce a multi-role pipeline (Proposer-Coder-Solver) that bridges abstract reasoning and visual grounding through executable visual synthesis. Across multiple base models and multimodal benchmarks, MM-Zero yields average reasoning gains and improves its own base model under a same-backbone control.

2 Methodology

In this section, we present the reinforcement learning with verifiable rewards (RLVR) framework (Wen et al., 2025) and the self-evolving reward pipeline for MM-Zero.

2.1 Preliminaries

RLVR is a training paradigm for VLMs applicable to domains where the correctness of model outputs can be verified. A rule-based verifier $v : X \rightarrow \{0, 1\}$ assigns a binary reward to each generation

x_i :

$$r_i = v(x_i) = \begin{cases} 1, & \text{if } x_i \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Such verifiable rewards are particularly effective in reasoning-intensive tasks (Wei et al., 2022), such as math and code generation (Zhao et al., 2025b), where correctness can be objectively evaluated. Beyond correctness, rewards can also encode additional criteria such as answer diversity or adherence to a desired output format.

We adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a practical RL algorithm that eliminates the need for a learned value function by instead computing relative rewards across multiple samples from the same prompt. Specifically, given a prompt p , the current policy $\pi_{\theta_{\text{old}}}$ generates a group of N responses $\{x_1, \dots, x_N\}$ with corresponding rewards $\{r_1, \dots, r_N\}$. These rewards are normalized within the group to yield response-level advantages:

$$\hat{A}_i = \frac{r_i - \text{mean}(r_1, \dots, r_N)}{\text{std}(r_1, \dots, r_N) + \varepsilon_{\text{norm}}}, \quad (2)$$

where $\varepsilon_{\text{norm}}$ is a small constant added for numerical stability. The policy is then updated by maximizing a clipped surrogate objective, regularized with a KL divergence term to constrain drift from the previous policy:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{N} \sum_{i=1}^N \min \left(\frac{\pi_{\theta}(x_i)}{\pi_{\theta_{\text{old}}}(x_i)} \hat{A}_i, \text{clip} \left(\frac{\pi_{\theta}(x_i)}{\pi_{\theta_{\text{old}}}(x_i)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_i \right) + \beta \text{KL}(\pi_{\theta} \parallel \pi_{\theta_{\text{old}}}) \quad (3)$$

By upweighting responses with positive relative advantages while penalizing excessive policy deviation, GRPO provides an effective mechanism for improving reasoning and generation quality in VLMs under the RLVR framework.

2.2 Training Pipeline

MM-Zero contains three distinct model agents evolved from the same base model using no additional data: a Proposer (π_P), a Coder (π_D), and a Solver (π_S), as illustrated in Figure 1. Each model is optimized sequentially via GRPO, with role-specific reward functions that form a closed training loop. During each role’s training phase, the other two roles **remain frozen**. We now describe the training pipeline for each role.

Proposer. The Proposer is trained to generate visual captions and questionanswer pairs. Zero-data does not mean unguided: the proposer is constrained by a structured prompt, seven fixed content types, required output fields, and downstream reward and filtering signals. At each training step, the Proposer is prompted to generate captions and questions across diverse visual domains, e.g., chart understanding, object and shape recognition, OCR, and visual mathematical reasoning. To compute rewards for the proposer, we initialize vLLM (Kwon et al., 2023) service ports for both the coder and solver. The coder receives the generated captions and questions and produces SVG code, which is rendered in parallel and converted to PNG images. Successfully rendered images, along with their associated easy and hard questions, are then forwarded to the solver to generate answers. The solver computes rewards for the proposer. We use a rollout size of 4 for the coder and 5 for the solver to reduce reward computation cost.

Coder. We use the most recent proposer checkpoint to generate approximately 4,000 caption and questionanswer pairs, which serve as training data for the coder. The coder is trained to generate SVG code that renders images from captions. Rendered images are sent to the solver to compute reward.

Solver. Training data for the solver is constructed using the most recent proposer and coder checkpoints. The proposer generates proposals, and the coder renders them into images. We retain only successfully rendered images and their associated questions as the training set for the solver.

Training data filtering. We apply stage-specific data filters to ensure training quality. For the coder, given a caption along with easy and hard questions and answers, we retain only examples where the rendering success rate across 4 rollouts falls between 0.25 and 0.75, excluding captions that are either trivially simple or too difficult to render. For the solver, we evaluate each successfully rendered image on both easy and hard questions, keeping only examples where easy-question accuracy exceeds 0.5 and hard-question accuracy falls between 0.27 and 0.75. This filtering ensures both stages are trained on examples of appropriate difficulty, neither trivially easy nor impossibly hard. It also makes instability self-limiting: weak roles yield fewer retained samples rather than low-quality supervision that propagates downstream.

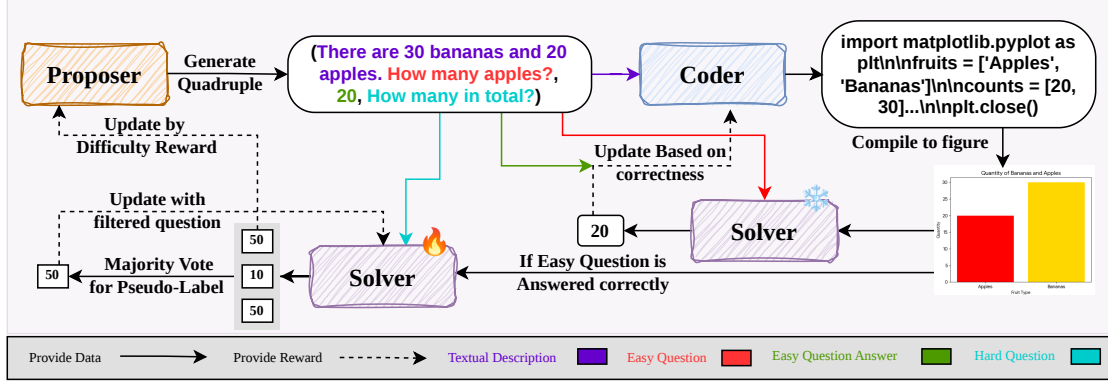


Figure 1: **Overview of the MM-Zero self-evolving framework.** The Proposer generates a quadruple (textual description, easy question, easy answer, hard question). The Coder converts the description into executable code to render a figure. The Solver first answers the easy question to verify semantic correctness and provide reward signals to update the Coder, then answers the hard question using majority voting to generate pseudo-labels for its own training while providing a difficulty reward to optimize the Proposer.

If post-filtering data becomes too sparse, increasing the rollout count is a simple stability knob.

2.3 Reward Formulations

In this section, we describe the reward design for each of the three roles.

2.4 Proposer Reward Formulation

The Proposer policy π_P generates a proposal Quadruple $x = (c, q_{\text{easy}}, a_{\text{easy}}, q_{\text{hard}})$, where c is a fine-grained textual description of a visual scene, q_{easy} is a simple question whose answer becomes obvious when the caption is successfully rendered into an image (used to verify rendering quality), and q_{hard} is a challenging question requiring multi-step reasoning over the rendered image (used to train and evaluate the Solver). Through training, the Proposer learns to generate richer captions and harder questions, pushing the Coder to produce more informative renderings and the Solver to perform deeper reasoning. To evaluate x , we use a multi-stage and hierarchical reward mechanism that validates formatting, solvability, and difficulty via the Solver π_S . The final reward $R(x)$ is defined as:

$$R_P(x) = \begin{cases} -1 & \text{if format invalid,} \\ \frac{1}{N} \sum_{i=1}^N \mathbb{1}_{\text{exec}}(C_i) \cdot (\min(R_{\text{solv}}(I_i), 0.5) + R_{\text{diff}}(I_i)) + r_{\text{ch}} + r_{\text{ct}} + r_{\text{div}} & \text{otherwise.} \end{cases} \quad (4)$$

Here, $\mathbb{1}_{\text{exec}}(C_i)$ indicates whether code sample C_i successfully renders an image I_i . The solvability score verifies whether the rendered image preserves the Proposer’s intent by asking the Solver

to answer the easy question:

$$R_{\text{solv}}(I_i) = \frac{1}{K} \sum_{k=1}^K \mathbb{1} \left(y_{\text{easy}}^{(i,k)} = a_{\text{easy}} \right) \in [0, 1] \quad (5)$$

We cap solvability at 0.5 so the Proposer must balance renderable/easy-verifiable images with genuinely difficult hard questions. For hard questions without gold labels, we use test-time reinforcement learning (TTRL) (Zuo et al., 2025) self-consistency: the Solver performs K rollouts and c_i measures agreement with the majority answer $\hat{y}_i$:

$$c_i = \frac{1}{K} \sum_{k=1}^K \mathbb{1} \left(y_{\text{hard}}^{(i,k)} = \hat{y}_i \right) \quad (6)$$

The difficulty reward applies the **Goldilocks** principle (Johnston, 1990), peaking when the Solver is uncertain:

$$R_{\text{diff}}(I_i) = \min(c_i, 1 - c_i) \in [0, 0.5] \quad (7)$$

The additional terms r_{ch} , r_{ct} , and r_{div} penalize overly easy hard questions, over-represented content types, and repetitive captions/questions, respectively. Here content type refers to one of seven fixed categories: data_chart, diagram, geometry, timeline, map, table, and other (see Appendix A). The total reward range is $[-1.0, 1.5]$. Detailed definitions are provided in Appendix B.

2.5 Coder Reward: Execution and Validity

The Coder π_D receives a caption c and generates code C . Its goal is to produce executable code that

faithfully represents the Proposer’s intent. The reward R_D is a weighted sum of execution status, semantic correctness, and task feasibility:

$$R_D(C) = R_{\text{render}} + R_{\text{solv}} + R_{\text{diff}} - \lambda_{\text{err}} \quad (8)$$

where:

- $R_{\text{render}} = \mathbb{1}_{\text{exec}}(C) \in \{0, 1\}$ encourages valid code generation.
- $R_{\text{solv}} \in [0, 1]$ is identical to the Proposer’s Solvability Score, ensuring the rendered image is solvable.
- $R_{\text{diff}} \in \{0, 1\}$ is identical to the Proposer’s Difficulty Score for the hard question.
- λ_{err} applies penalties for syntax errors and rendering issues: -0.1 if rendering fails and -0.05 if the code has a syntax error.

2.6 Solver Reward: Test-Time Reinforcement Learning

The Solver policy π_S is trained on hard questions (I, q_{hard}) generated by the proposer. Since ground truth labels are unavailable for these generated questions, we use TTRL via majority voting. To keep these silver labels informative, solver training uses only filtered examples whose easy-question accuracy exceeds 0.5 and whose hard-question accuracy remains in the non-degenerate range $[0.27, 0.75]$.

For a given input, the Solver generates K independent reasoning paths. We identify the **silver answer** $\bar{y}$ via majority vote: $\bar{y} = \text{Mode}(\{y_1, \dots, y_K\})$. The reward for the k -th response is defined as a weighted sum of answer accuracy and structural validity:

$$R_S(y_k) = \alpha \cdot R_{\text{acc}}(y_k, \bar{y}) + (1 - \alpha) \cdot R_{\text{fmt}}(y_k) \quad (9)$$

where $\alpha = 0.9$. R_{acc} checks whether the extracted `\boxed{\}` answer matches the majority-vote silver answer, while R_{fmt} enforces the required reasoning-plus-boxed-answer structure. Full definitions are in Appendix B.

3 Experiment

We use Qwen3-VL-4B-Instruct, 8B (Yang et al., 2025) and MiMo-VL-7B-SFT (Xiaomi, 2025) as our base models. Training proceeds in an iterative

loop over three roles (proposer, coder, and solver) saving a checkpoint for each role every 20 steps, for a total of 60 steps. The training details and training time details are in Appendix D.¹ Unless otherwise noted, all reward coefficients and training hyperparameters are shared across backbones.

3.1 Solver Evaluation Data

We evaluate the Solver on a diverse set of VLM benchmarks covering structured visual reasoning, including chart understanding and visual mathematics, as well as benchmarks containing natural-image understanding tasks.

Natural Visual Reasoning. We evaluate whether training on synthetically generated visuals preserves, and potentially improves, the Solver’s ability to understand natural images. We use MMMU (Yue et al., 2024a), MMMU-Pro (Yue et al., 2024b), MM-Vet (Yu et al., 2023), and HallusionBench (Guan et al., 2024), which include real-world images and test object recognition, visual grounding, hallucination, and multimodal reasoning.

Structured Visual Reasoning. We also evaluate on structured visual understanding tasks, including chart understanding, visual geometry, mathematical reasoning, and numerical reasoning. Specifically, we use MathVerse (Zhang et al., 2024), MathVision (Bleeker et al., 2024), MathVista (Lu et al., 2024), VisNumBench (Weng et al., 2025), and ChartQA (Masry et al., 2022). These benchmarks test rigorous quantitative and geometric reasoning over structured visual inputs, similar to the capabilities targeted by our SVG-based generated images.

3.2 Main Results

We use VLMEvalKit (Duan et al., 2024), a standard toolkit for VLM evaluation.² We present the results of the solver in Table 1.

The solver achieves average gains across benchmarks over training iterations compared to the base models. Results in Table 1 show that across different model sizes, average accuracy on VLM

¹All training is conducted on 8 RTX 6000/Pro 96GB GPUs.

²Due to cost constraints, we use Qwen-3-32B (Yang et al., 2025) as the judge in place of the GPT-4o judge used in the Qwen3-VL official reports. This may lead to minor variations from the reported Qwen results.

Table 1: The average accuracy of MM-Zero improves over training iterations, although some individual benchmarks remain flat or fluctuate. Avg. is computed over the 8 benchmarks with available reported scores. The first block lists published prior self-evolution baselines; the second block is our same-backbone control on Qwen2.5-VL-7B-Instruct.

Methods	Natural Visual Reasoning				Structured Visual Reasoning					Avg.
	MMMU	MMMU -Pro	MM -Vet	Hallusion Bench	ChartQA	Math Verse	Math -Vision	MathVista	VisNum Bench	
<i>Prior VLM Self-Evolution Methods (Qwen2.5-VL-7B-Instruct Backbone)</i>										
VisPlay (He et al., 2025)	38.3	-	46.3	92.3	-	39.1	31.2	-	32.6	-
Vision-Zero (Wang et al., 2025c)	-	-	-	-	86.3	52.2	28.0	72.4	-	-
V-Zero (Wang et al., 2026)	58.6	-	-	-	-	42.6	27.0	69.2	-	-
<i>Same-Backbone Control (Qwen2.5-VL-7B-Instruct)</i>										
Base Model (published)	58.6	-	67.1	52.9	87.3	43.1	25.1	68.2	43.8	55.8
MM-Zero (Solver v3)	60.8	-	69.6	55.2	88.6	45.7	27.6	71.2	46.2	58.1
<i>Qwen3-VL-4B-Instruct</i>										
Base Model	53.4	36.5	71.1	51.4	79.1	52.1	31.6	61.3	45.3	53.5
MM-Zero (Step 20, Iter 1)	55.9	44.6	71.5	67.8	79.8	49.8	36.9	66.0	49.6	58.0
MM-Zero (Step 40, Iter 2)	56.8	45.1	71.8	69.1	79.6	49.4	37.1	64.7	49.1	58.1
MM-Zero (Step 60, Iter 3)	57.7	45.3	72.1	69.1	80.3	50.0	36.7	65.4	49.9	58.5
<i>Qwen3-VL-8B-Instruct</i>										
Base Model	69.6	55.9	76.1	61.1	89.6	62.1	53.9	77.2	47.7	65.9
MM-Zero (Step 20, Iter 1)	70.4	59.3	76.0	71.9	90.0	62.8	54.6	81.7	53.2	68.9
MM-Zero (Step 40, Iter 2)	71.8	61.3	75.4	71.3	90.4	64.2	55.1	83.1	54.5	69.7
MM-Zero (Step 60, Iter 3)	72.0	63.0	77.2	71.7	90.4	65.1	55.2	84.0	55.0	70.4
<i>MiMo-VL-7B-SFT</i>										
VisPlay*	28.2	-	56.9	74.6	-	46.0	29.4	-	52.7	-
Base Model	64.6	45.2	75.6	70.8	92.9	67.1	57.9	71.8	44.7	65.6
MM-Zero (Step 20, Iter 1)	68.4	48.4	75.3	73.0	95.5	55.6	62.5	78.7	48.0	67.3
MM-Zero (Step 40, Iter 2)	68.9	48.7	75.8	73.2	95.0	56.0	61.9	77.5	48.3	67.3
MM-Zero (Step 60, Iter 3)	68.9	48.7	76.9	75.0	95.0	56.0	62.7	78.9	48.5	67.8

benchmarks improves with additional training iterations, although some individual benchmarks remain flat or fluctuate. Specifically, for Qwen3-VL-8B-Instruct, our solver at iteration 3 achieves an average score of 70.4%, compared to 65.9% for the base model, an improvement of 4.5 points without any external data. Qwen3-VL-4B-Instruct and MiMo-VL-7B-SFT also improve on average, even though MathVerse remains below the base model for those two backbones. Matched-decoding controls in Appendix 5 rerun the Qwen3-VL-8B base and iteration-3 solver under each other’s temperatures and change the average by at most 0.3 points, leaving the main training gap essentially unchanged. These results demonstrate the effectiveness of MM-Zero at improving VLM reasoning without any external data, pointing toward a promising direction for self-evolving multimodal models.

MM-Zero generalizes across different models and model sizes. As shown in Table 1, the average score for the 4B model increases from 53.5 to 58.5, while MiMo-VL-7B-SFT, a different model family from Qwen-VL, improves from 65.6 to

67.8. In addition, the same-backbone Qwen2.5-VL-7B-Instruct control in Table 1 improves from 55.8 to 58.1 average, showing that the gain is attributable to the method rather than only to a stronger backbone.

Coder render success rate improves throughout training. As shown in Figure 2, both render success rate and image solvability steadily improve during training. This indicates that the Coder learns to generate more executable SVGs, while the rendered images become increasingly faithful to the Proposer’s intent and sufficient for answering the easy verification questions.

3.3 Qualitative Analysis

We present case studies for Qwen3-VL-8B in Figure 3, showing proposer outputs and rendered images across training iterations. For each iteration, we sample 10 successfully rendered images and manually inspect them alongside their corresponding questions. For the base model, generated figures suffer from cluttered layouts with overlapping elements, making them nearly unreadable and leading to low solvability rewards. By Iter-

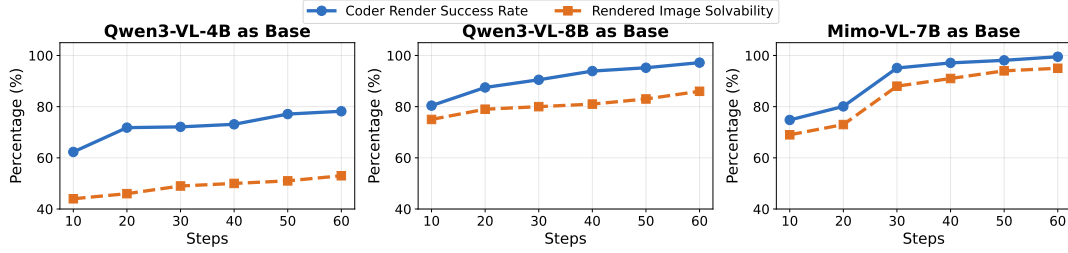


Figure 2: Throughout training, the coder’s image rendering success rate steadily improves, indicating stronger code generation ability. Additionally, image solvability also increases, meaning the rendered images become more faithful to the original and contain sufficient information to answer the question.

ation 1, visual organization improves and questions begin to demand greater reasoning, though answers are often embedded directly in the image as annotations, making them trivially easy, which is a pattern our difficulty score penalizes. By Iteration 2, visuals become cleaner and hard questions require multi-step reasoning rather than simple value extraction. By Iteration 3, layouts are more polished and questions demand genuine compositional reasoning: for instance, reading an absolute value from the y-axis, then applying a percentage annotation to compute a derived quantity.

4 Ablation Study

In this section, we conduct a reward ablation study to examine how individual reward components affect overall model performance. Prior self-evolving frameworks, such as R-Zero (Huang et al., 2025a) for LLMs and Visplay (He et al., 2025) for VLMs, employ only two roles and rely on difficulty and question diversity rewards. In contrast, MM-Zero introduces a three-role framework that includes a coder responsible for rendering images, necessitating additional reward designs to ensure the quality of generated code and rendered images. We identify two key reward components unique to our framework and study their contributions by removing each one individually and retraining Qwen3-VL-8B-Instruct.

Solvability and Difficulty Balance. Referring to Equation (4), the proposer reward includes the term $(\min(R_{\text{solv}}, 0.5) + R_{\text{diff}})$, where solvability is capped at 0.5. This cap prevents the proposer from being overly rewarded for generating easy, trivially solvable instances, thereby encouraging a balance between solvability and difficulty. In our ablation, we remove this cap and replace the term with $(R_{\text{solv}}(I_i) + R_{\text{diff}}(I_i))$. We train the ablated

model using identical hyperparameters for three iterations.

Content Diversity. Also in Equation (4), the factor r_{ct} measures the diversity of visual content types in the generated captions. We include this term to incentivize the proposer to produce captions spanning a wider range of fixed visual content categories (data_chart, diagram, geometry, timeline, map, table, and other), rather than repeatedly generating instances of a single type.

Removing solvability balance leads to reward hacking by the coder. Without the solvability cap, the ablated model improves from 65.9% to only 68.0%, smaller than full MM-Zero in Table 1. Because uncapped solvability can dominate the difficulty term, the model learns a shortcut: generating easy-to-solve renderings and sometimes embedding answers directly in the image. The capped reward mitigates this reward-hacking pattern by forcing solvability and difficulty to contribute more evenly.

Lack of diversity leads to overfitting to a narrow set of visual problem types. Without the content diversity reward, performance initially rises from 65.9% to 67.4% but then declines to 65.5% by iteration 3. Manual inspection shows that generated samples increasingly collapse toward easy-to-render visual types such as histograms. Thus, diversity regularization prevents mode collapse and supports generalization across the broader benchmark suite.

Coder evolution is important for sustained gains. To isolate the three-role contribution, we freeze the coder and train only the proposer and solver on Qwen3-VL-4B-Instruct. As shown in Table 3, this two-role variant improves the average by only +0.7, +0.4, and +0.6 points across the

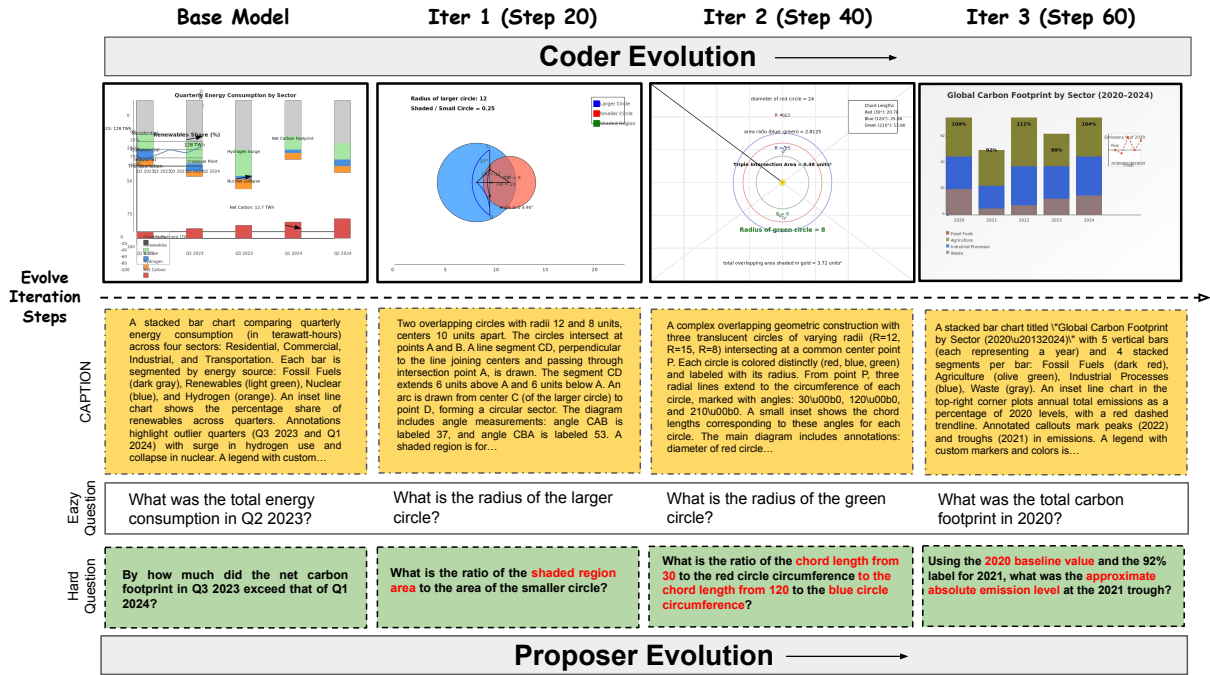


Figure 3: Evolution of the proposer and coder. This figure presents examples of questions generated by the self-evolving Vision-Language model. As the training steps progress, the visual context becomes more intricate, and the model generates increasingly challenging questions. While the upper bound of complexity grows, foundational and simple questions are still preserved. The questions are formulated by Qwen3-VL-8B-Instruct.

Table 2: Ablation study on the solvability–difficulty balance and content diversity rewards. We group benchmarks into natural and structured visual reasoning. Removing solvability capping slows improvement, while removing content diversity causes overfitting and declining accuracy.

Methods	Natural Visual Reasoning				Structured Visual Reasoning					Avg.
	MMMU	MMMU-Pro	MM-Vet	Hallusion Bench	ChartQA	Math Verse	Math -Vision	MathVista	VisNum Bench	
<i>Qwen3-VL-8B-Instruct</i>										
Base Model	69.6	55.9	76.1	61.1	89.6	62.1	53.9	77.2	47.7	65.9
<i>w/o Solvability & Difficulty Balance</i>										
Step 20, Iter 1	71.7	54.5	80.7	59.5	90.4	62.5	54.5	77.1	50.1	66.8
Step 40, Iter 2	72.3	54.7	79.3	60.7	89.8	64.5	58.7	77.9	54.4	68.0
Step 60, Iter 3	70.4	56.1	78.8	58.9	91.1	64.9	59.4	77.6	54.8	68.0
<i>w/o Content Diversity</i>										
Step 20, Iter 1	73.3	56.7	77.5	61.4	89.9	64.0	55.5	76.9	51.0	67.4
Step 40, Iter 2	71.9	55.9	77.9	60.9	86.1	64.5	57.2	77.3	51.8	67.1↓
Step 60, Iter 3	69.1	54.3	76.1	60.5	81.8	63.6	57.5	75.4	51.5	65.5↓

Table 3: Frozen-coder two-role ablation on Qwen3-VL-4B-Instruct. The two-role variant improves only slightly over the base model and remains well below the full three-role run on the same backbone.

Checkpoint	MMMU	MMMU-Pro	MM-Vet	HallusionBench	ChartQA	MathVerse	MathVision	MathVista	VisNumBench	Avg.
Base (paper)	53.4	36.5	71.1	51.4	79.1	52.1	31.6	61.3	45.3	53.5
solver_v1	55.6	37.2	74.2	51.8	78.4	51.7	32.1	60.1	47.1	54.2
solver_v2	54.3	36.8	72.7	51.4	78.5	52.1	31.6	61.0	46.5	53.9
solver_v3	54.2	37.4	72.6	52.7	78.8	51.2	31.5	61.1	47.1	54.1

three iterations, far below the +4.5, +4.6, and +5.0 gains of full MM-Zero on the same backbone.

This suggests that coder evolution is not merely auxiliary; it is needed to keep the loop generating richer and harder visual tasks.

Table 4: Ablation without r_{eh} on Qwen3-VL-8B-Instruct. Removing the easy-hard penalty improves over the base model early on but weakens later-stage stability compared with full MM-Zero.

Method	MMMU	MMMU-Pro	MM-Vet	HallusionBench	ChartQA	MathVerse	MathVision	MathVista	VisNumBench	Avg.
Base Model	69.6	55.9	76.1	61.1	89.6	62.1	53.9	77.2	47.7	65.9
w/o r_{eh} - Step 20 (Iter 1)	70.0	58.9	76.4	70.8	89.8	62.4	54.9	80.9	53.0	68.6
w/o r_{eh} - Step 40 (Iter 2)	71.6	60.8	77.2	71.0	90.2	64.2	55.0	83.0	54.3	69.7
w/o r_{eh} - Step 60 (Iter 3)	71.0	60.2	76.5	70.4	89.9	63.5	54.6	82.0	53.8	69.1

The easy-hard penalty stabilizes later iterations. We additionally remove the easy-hard penalty r_{eh} while keeping the other reward terms fixed. As shown in Table 4, the model improves over the base model early on but peaks at iteration 2 and declines at iteration 3, whereas full MM-Zero continues improving through the final checkpoint.

Thus, r_{eh} is a targeted regularizer that discourages the proposer from drifting toward overly easy hard questions and stabilizes later training stages.

5 Related Work

5.1 Reinforcement Learning with Verifiable Rewards

This paradigm has demonstrated broad success in domains where objective correctness can be programmatically evaluated, ranging from traditional mathematical and code generation tasks (Guo et al., 2025; Wang et al., 2025a) to complex multimodal challenges (Huang et al., 2025b; Wang et al., 2025b; Li et al., 2025b) and structured data environments (Shi et al., 2025; Li et al., 2026b; Merrill et al., 2026). To effectively leverage these deterministic feedback signals and enable complex behaviors like MM-Zero, concurrent research is actively refining algorithmic techniques. Emphasis has shifted towards optimizing policy learning under verifiable constraints, utilizing robust frameworks like DAPO (Yu et al., 2025) and VAPO (Yue et al., 2025), alongside high-entropy guided optimization (Dai et al., 2025; Wang et al., 2025d) to encourage diverse exploration and prevent premature convergence within sparse, rule-based reward landscapes.

5.2 Self-Evolution in Vision-Language Models

The recent success of self-evolution in LLMs (Xiang et al., 2026; Yang et al., 2026), with minimal or no data, has begun to extend into the VLM domain (He et al., 2025; Wang et al., 2025c; Tong

et al., 2025; Wei et al., 2025). However, unlike the single-modality setting, self-evolution in multimodal models requires generating or sourcing additional visual data for training, making the problem fundamentally more challenging (Chen et al., 2023; Li et al., 2026c; Zheng et al., 2025). Existing VLM self-evolution approaches, such as VisPlay (He et al., 2025), adapt the LLM *challenger-solver* pipeline to multimodal tasks but strictly require a collection of seed images to bootstrap training, which can be costly to curate and filter. Other concurrent works, including Evolmm (Thawakar et al., 2025) and V-Zero (Wang et al., 2026), explore multimodal self-improvement with minimal annotation but remain conditioned on pre-existing static image datasets. While these methods reduce the need for human-annotated labels, they remain constrained by a fixed image corpus, preventing the synthesis of progressively harder or more diverse visual scenes that could continuously challenge the model’s perception and reasoning (He et al., 2025; Thawakar et al., 2025; Wang et al., 2026). Our **MM-Zero** framework addresses this limitation by introducing a generative Coder role that programmatically renders visual content, enabling true zero-data self-evolution.

6 Conclusion

We present MM-Zero, a self-evolving framework that trains VLMs starting from zero data. Unlike prior self-evolving approaches, MM-Zero instantiates more than two agents from the same base model, each fulfilling a distinct role: Proposer, Coder, and Solver. Our results demonstrate the scalability and effectiveness of multi-model multimodal evolution without any external training data: with carefully designed reward functions tailored to each role, sequential training across roles improves average benchmark performance across iterations. Future work can extend tool usage beyond code generation to produce more diverse diagrams, plots, and 3D renderings, creating richer training data for capabilities such as 3D spatial reasoning.

Limitations

MM-Zero demonstrates that VLMs can self-evolve by generating their own visual data and questions. Our results across multiple model sizes show that stronger base models benefit more from self-evolution, as they produce higher-quality visual code. A key limitation is that SVG-rendered images are inherently structured and may not fully capture the fine-grained appearance, texture, and complexity of natural images. This scope is deliberate: structured SVGs provide exact, verifiable supervision, which is what makes zero-data self-evolution feasible in the first place. Even under this restricted setting, the resulting gains transfer to several natural-image benchmarks. Future work can extend this framework to multi-backbone systems, for example by incorporating diffusion models or stronger visual generators to synthesize more diverse and naturalistic images while retaining the closed-loop self-evolving paradigm.

Ethical Considerations

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here. In addition, we use AI solely for coding, grammar check and corrections.

References

- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhi-fang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Ming-sheng Li, and 45 others. 2025. [Qwen3-vl technical report](#). *Preprint*, arXiv:2511.21631.
- Maurits Bleeker, Mariya Hendriksen, Andrew Yates, and Maarten de Rijke. 2024. [Demonstrating and reducing shortcuts in vision-language representation learning](#). *Preprint*, arXiv:2402.17510.
- Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. 2023. [Sharegpt4v: Improving large multi-modal models with better captions](#). *Preprint*, arXiv:2311.12793.
- Jeff Clune. 2019. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*.
- Runpeng Dai, Linfeng Song, Haolin Liu, Zhenwen Liang, Dian Yu, Haitao Mi, Zhaopeng Tu, Rui Liu, Tong Zheng, Hongtu Zhu, and 1 others. 2025. Cde: Curiosity-driven exploration for efficient reinforcement learning in large language models. *arXiv preprint arXiv:2509.09675*.
- Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. [CARLA: An open urban driving simulator](#). In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR.
- Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, and 1 others. 2024. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201.
- Hao Fei, Yuan Yao, Zhuosheng Zhang, Fuxiao Liu, Ao Zhang, and Tat-Seng Chua. 2024. From multi-modal llm to human-level ai: Modality, instruction, reasoning, efficiency and beyond. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024): Tutorial Summaries*, pages 1–8.
- Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. 2025. [Video-r1: Reinforcing video reasoning in mllms](#). *Preprint*, arXiv:2503.21776.
- Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. 2024. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. 2025. Visplay: Self-evolving vision-language models from images. *arXiv preprint arXiv:2511.15661*.
- Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025a. [R-zero: Self-evolving reasoning llm from zero data](#).
- Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, and 1 others. 2025b. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*.

- Sean Johnston. 1990. [The goldilocks principle](#). *Physics World*, 3(6):76.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Zongxia Li, Hongyang Du, Dawei Liu, Xiyang Wu, Lantao Yu, Jingxi Chen, Fuxiao Liu, Xiaomin Wu, Jing Xie, Chengsong Huang, Yicheng He, and Guangyao Shi. 2026a. Multimodal video generation models with audio: Present and future.
- Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, and 1 others. 2026b. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. *arXiv preprint arXiv:2607.08964*.
- Zongxia Li, Dawei Liu, Fuxiao Liu, Yuhang Zhou, Xiyang Wu, Jingxi Chen, Jing Xie, Xiaomin Wu, and Lichao Sun. 2026c. Comfyclaw: Self-evolving skill harnesses for image generation workflows. *arXiv preprint arXiv:2607.01709*.
- Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem, and Guangyao Shi. 2025a. [A survey of state of the art large vision language models: Alignment, benchmark, evaluations and challenges](#). In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1578–1597.
- Zongxia Li, Wenhao Yu, Chengsong Huang, Rui Liu, Zhenwen Liang, and 1 others. 2025b. Self-rewarding vision-language model via reasoning decomposition. *arXiv preprint arXiv:2508.19652*.
- Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilya Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. 2025. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*.
- Fuxiao Liu, Kevin Lin, Linjie Li, Jianfeng Wang, Yaser Yacoob, and Lijuan Wang. 2023a. Mitigating hallucination in large multi-modal models via robust instruction tuning. *arXiv preprint arXiv:2306.14565*.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023b. [Visual instruction tuning](#). *Preprint*, arXiv:2304.08485.
- Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. [On LLMs-driven synthetic data generation, curation, and evaluation: A survey](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11065–11082, Bangkok, Thailand. Association for Computational Linguistics.
- Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*.
- Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. [ChartQA: A benchmark for question answering about charts with visual and logical reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, Dublin, Ireland. Association for Computational Linguistics.
- Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, and 1 others. 2026. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pages 40903–40986.
- Jürgen Schmidhuber. 2007. Gödel machines: Fully self-referential optimal universal self-improvers. In *Artificial general intelligence*, pages 199–226. Springer.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, and 1 others. 2025. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*.
- David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. [Mastering the game of Go with deep neural networks and tree search](#). *Nature*, 529(7587):484–489.
- Omkar Thawakar, Shravan Venkatraman, Ritesh Thawakar, Abdelrahman Shaker, Hisham Cholakkal, Rao Muhammad Anwer, Salman Khan, and Fahad Khan. 2025. Evolmm: Self-evolving large multimodal models with continuous rewards. *arXiv preprint arXiv:2511.16672*.
- Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. [Domain randomization for transferring deep neural networks from simulation to the real world](#). In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

- Jingqi Tong, Jixin Tang, Hangcheng Li, Yurong Mou, Ming Zhang, Jun Zhao, Yanbo Wen, Fan Song, Jiahao Zhan, Yuyang Lu, Chaoran Tao, Zhiyuan Guo, Jizhou Yu, Tianhao Cheng, Zhiheng Xi, Changhao Jiang, Zhangyue Yin, Yining Zheng, Weifeng Ge, and 5 others. 2025. [Game-rl: Synthesizing multi-modal verifiable game data to boost vlms' general reasoning](#). *Preprint*, arXiv:2505.13886.
- Han Wang, Yi Yang, Jingyuan Hu, Minfeng Zhu, and Wei Chen. 2026. V-zero: Self-improving multimodal reasoning with zero annotation. *arXiv preprint arXiv:2601.10094*.
- Haozhe Wang, Long Li, Chao Qu, Fengming Zhu, Weidi Xu, and 1 others. 2025a. To code or not to code? adaptive tool integration for math language models via expectation-maximization. *arXiv preprint arXiv:2502.00691*.
- Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and 1 others. 2025b. VI-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *arXiv preprint arXiv:2504.08837*.
- Qinsi Wang, Bo Liu, Tianyi Zhou, Jing Shi, Yueqian Lin, Yiran Chen, Hai Helen Li, Kun Wan, and Wentian Zhao. 2025c. [Vision-zero: Scalable vlm self-improvement via strategic gamified self-play](#). *Preprint*, arXiv:2509.25541.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, and 1 others. 2025d. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA. Curran Associates Inc.
- Lai Wei, Yuting Li, Chen Wang, Yue Wang, Linghe Kong, Weiran Huang, and Lichao Sun. 2025. First sft, second rl, third upt: Continual improving multimodal llm reasoning via unsupervised post-training. *arXiv preprint arXiv:2505.22453*.
- Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, and 1 others. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.
- Tengjin Weng, Jingyi Wang, Wenhao Jiang, and Zhong Ming. 2025. Visnumbench: Evaluating number sense of multimodal large language models. *arXiv preprint arXiv:2503.14939*.
- Zhishang Xiang, Chengyi Yang, Zerui Chen, Zhimin Wei, Yunbo Tang, Zongpei Teng, Zexi Peng, Zongxia Li, Chengsong Huang, Yicheng He, Chang Yang, Xinrun Wang, Xiao Huang, Qinggang Zhang, and Jinsong Su. 2026. [A systematic survey of self-evolving agents: From model-centric to environment-driven co-evolution](#).
- Zhishang Xiang, Chengyi Yang, Zerui Chen, Zhimin Wei, Yunbo Tang, Zongpei Teng, Zexi Peng, Zongxia Li, Chengsong Huang, Yicheng He, and 1 others. A systematic survey of self-evolving agents: From model-centric to environment-driven co-evolution.
- LLM-Core-Team Xiaomi. 2025. [Mimo-vl technical report](#). *Preprint*, arXiv:2506.03569.
- An Yang, Anfeng Li, and Baosong Yang et al. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Chengyi Yang, Zhishang Xiang, Yunbo Tang, Zongpei Teng, Chengsong Huang, Fei Long, Yuhan Liu, and Jinsong Su. 2026. Ttcs: Test-time curriculum synthesis for self-evolving. *arXiv preprint arXiv:2601.22628*.
- Yuqing Yang, Yan Ma, and Pengfei Liu. 2024. Weak-to-strong reasoning. *arXiv preprint arXiv:2407.13647*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2023. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*.
- Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruofei Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, and 1 others. 2024a. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.
- Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, and 1 others. 2024b. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*.
- Yu Yue, Yufeng Yuan, Qiyang Yu, Xiaochen Zuo, Ruofei Zhu, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.
- Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. 2026a. [Dr. zero: Self-evolving search agents without training data](#). *Preprint*, arXiv:2601.07055.

- Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. 2026b. Dr. zero: Self-evolving search agents without training data. *arXiv preprint arXiv:2601.07055*.
- Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. 2024. [Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems?](#) *Preprint*, arXiv:2403.14624.
- Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025a. [Absolute zero: Reinforced self-play reasoning with zero data.](#) *Preprint*, arXiv:2505.03335.
- Yulai Zhao, Haolin Liu, Dian Yu, Sunyuan Kung, Meijia Chen, Haitao Mi, and Dong Yu. 2025b. One token to fool llm-as-a-judge. *arXiv preprint arXiv:2507.08794*.
- Qi Zheng, Haozhi Wang, Zihao Liu, Jiaming Liu, Zhi-jian Hao, Bu Chen, Min Li, Rui Wan, Peiye Liu, Yanheng Lu, and 1 others. 2025. Unicorn: Unified neural image compression with one number reconstruction. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8626–8635.
- Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, and 1 others. 2025. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

Appendix

A Prompt Templates

We provide the prompt templates and training hyperparameters for reproducibility.

A.1 Proposer

The Proposer prompt instructs the model to act as a Visual Content Designer and output six XML blocks: `content_type`, `caption`, `easy_question`, `easy_answer`, `hard_question`, and `hard_answer`. Allowed `content_type` values are `data_chart`, `diagram`, `geometry`, `timeline`, `map`, `table`, and `other`. The full prompt is shown below.

Proposer Prompt Template

Role: You are an expert Visual Content Designer who creates rich, complex data visualizations and diagram specifications using SVG. Your goal is to design visualizations that are visually interesting, data-dense, and require genuine reasoning to interpret.

Input: `{{ content | trim }}`

Output — exactly six XML blocks and nothing else:

```
<content_type>exactly one of: data_chart,
diagram, geometry, timeline, map, table,
other</content_type>
<caption>a rich, detailed specification
</caption>
<easy_question>a simple question directly
readable from the image</easy_question>
<easy_answer>the answer</easy_answer>
<hard_question>a challenging reasoning
question</hard_question>
<hard_answer>the answer</hard_answer>
```

Complexity requirements for captions — include at least three of: multiple data series, annotations, secondary panel, colors/markers, derived values, non-trivial patterns, geometric constructions.

Hard question constraints: Must require multi-step reasoning; must force the reader to extract at least one value from the visualization (do *not* state all values in the question).

Answer format: single number, word, or short phrase (e.g. "42", "Q1", "blue").

A.2 CodeGen

The CodeGen prompt instructs the model to generate raw SVG markup given a caption and questions. The rendered image must contain the data needed to answer the Easy Question. Full prompt:

SVG Generator Prompt Template

Input: `{{ content | trim }}`

You are an SVG code generator for data visualizations. You will be given a chart description (caption) and questions with short answers.

Critical: The rendered image **must** contain the data needed to answer the Easy Question with the exact Easy Answer provided.

Write raw SVG markup (starting with `<svg ...>`). Do **not** write Python code.

SVG guidelines: use `viewBox`; use `<text>` for labels; `font-size` ≥ 12 px; distinct colors; self-contained. Wrap your SVG in ```svg ... ``` code fences.

A.3 Solver

The Solver prompt instructs the model to think step-by-step in `<think>` tags and give the final answer in `\boxed{}`. Full prompt:

Solver Prompt Template

Input:

```
{% if '<image>' not in content %}
<image>{% endif %}{{ content | trim }}
```

Look at the image carefully and answer the question. First, think step by step inside `<think> ... </think>` tags. Then, give your final answer inside `\boxed{}` as a single number, single word, or short phrase only (e.g. `\boxed{42}`, `\boxed{blue}`, `\boxed{Q1}`) — no units, no full sentences.

A.4 Evaluation with VLMEvalKit

We evaluate all models using VLMEvalKit (Duan et al., 2024) on 9 benchmarks: ChartQA-test, Hal-lusionBench, MM-Vet, MMMU (val), MMMU-Pro (10-option and Vision), MathVerse-MINI (Vision-Only), MathVision, and MathVista-mini. Evaluation follows a two-phase pipeline: (1) inference via a vLLM (Kwon et al., 2023) OpenAI-compatible server with 32-way concurrent requests, and (2) scoring via `-mode eval -reuse` against a judge LLM served on a separate vLLM endpoint. We use Qwen3-VL-32B as the judge. Unless otherwise noted, the main-paper numbers use the original evaluation settings: baselines use near-greedy decoding (temperature=0.01), while solver checkpoints use diverse sampling (temperature=0.7 for Qwen3-VL and 0.6 for MiMo-VL-7B). Scoring is benchmark-dependent: rule-based exact-match for ChartQA and MMMU variants, and LLM-judge-based answer extraction for math and free-form benchmarks.

Table 5: Matched-decoding control for Qwen3-VL-8B-Instruct at Iteration 3. Rows marked “paper” use the settings reported in the main paper; rows marked “matched” rerun the same checkpoint under the other temperature. The average changes by at most 0.3 points, indicating that the main gap is not driven by decoding alone.

Setting	Decoding	MMMU	MMMU-Pro	MM-Vet	HallusionBench	ChartQA	MathVerse	MathVision	MathVista	VisNumBench	Avg.
Base (paper)	$T=0.01$	69.6	55.9	76.1	61.1	89.6	62.1	53.9	77.2	47.7	65.9
Base (matched to solver)	$T=0.7$	69.4	55.9	75.4	61.4	89.6	61.3	54.2	77.5	45.7	65.6
MM-Zero, Iter 3 (paper)	$T=0.7$	72.0	63.0	77.2	71.7	90.4	65.1	55.2	84.0	55.0	70.4
MM-Zero, Iter 3 (matched to base)	$T=0.01$	72.3	63.0	77.0	72.1	90.4	65.5	54.9	83.6	55.7	70.5

Training Parameters

Table 6: Main script parameters.

Param.	Val.	Description
GPU_MEM	80	GPU mem. (GB)
TRAIN_STEPS	10	Steps/model/iter.
NUM_ITER	3	Self-Evolve iter.
PROP_PER_GPU	413	Prop./GPU (P/CG)
PROP_PER_S	625	Prop./GPU (Solv.)
PROP_ROLL_BS	18	Prop. roll. BS
GLOBAL_BS_P	18	Prop. global BS
ROLLOUT_N	4	Rolls/prop. (P/CG)
SOLVER_N_R	5	Solver rollouts
ROLLOUT_BS	320	CG roll. BS
ROLL_BS_S	512	Solver roll. BS
GLOBAL_BS_S	64	Solver global BS

Per-Role Config (96 GB)

Table 7: Per-role training configuration.

Parameter	Proposer	Coder	Solver
GPUs	3	4	8
Max prompt len.	4096	4096	8192
Max resp. len.	2048	4096	4096
Roll. batch size	18	256	512
Global batch size	18	64	64
Learning rate	1×10^{-6}		
Weight decay	1×10^{-2}		
Roll. temp.	1.0	0.7	1.0
Roll. top_p	0.99	0.95	0.99
Rollout n	4	4	8
Tensor parallel	1	1	2

All reward coefficients and training hyperparameters are shared across Qwen3-VL-4B-Instruct, Qwen3-VL-8B-Instruct, and MiMo-VL-7B-SFT; we do not tune them per backbone.

B Reward Details

Proposer penalties and diversity terms. For the easy-hard penalty, let $\mathcal{I}$ be the set of successfully rendered images. We penalize proposals whose average hard-question difficulty is below δ_{eh} :

$$r_{\text{eh}} = \begin{cases} -\lambda_{\text{eh}} & \text{if } \frac{1}{|\mathcal{I}|} \sum_{I_i \in \mathcal{I}} R_{\text{diff}}(I_i) < \delta_{\text{eh}}, \\ 0 & \text{otherwise,} \end{cases} \quad (10)$$

where $\delta_{\text{eh}} = 0.15$ and $\lambda_{\text{eh}} = 0.3$. For content-type diversity, let f_t be the fraction of proposals in the

batch with the same content type as proposal x :

$$r_{\text{ct}} = \begin{cases} -\lambda_{\text{ct}} \cdot \frac{f_t - \phi}{1 - \phi} & \text{if } f_t > \phi, \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

where $\phi = 0.5$ and $\lambda_{\text{ct}} = 0.15$. For caption/question diversity, we cluster captions, easy questions, and hard questions using agglomerative clustering on pairwise BLEU distances. Let $s_x^{(\text{cap})}$, $s_x^{(\text{eq})}$, and $s_x^{(\text{hq})}$ be the cluster shares for proposal x , and let $u = 1/M$ be the uniform share for a batch of M valid proposals:

$$r_{\text{div}} = -\text{clip}\left((w_c(s_x^{(\text{cap})} - u) + w_e(s_x^{(\text{eq})} - u) + w_h(s_x^{(\text{hq})} - u)) \cdot M \cdot \lambda_{\text{div}}, -\lambda_{\text{div}}, \lambda_{\text{div}}\right) \quad (12)$$

where $w_c = 0.45$, $w_e = 0.20$, $w_h = 0.35$, and $\lambda_{\text{div}} = 0.5$. Proposals in small clusters receive a bonus, while those in large clusters are penalized.

Solver reward components. We extract the final answer $\hat{y}_k$ from the `\boxed{\dots}` content in response y_k and compare it against the majority-vote silver answer $\bar{y}$:

$$R_{\text{acc}}(y_k, \bar{y}) = \mathbb{1}(\hat{y}_k = \bar{y}). \quad (13)$$

The format reward enforces the required reasoning trace and boxed final answer:

$$R_{\text{fmt}}(y_k) = \mathbb{1}(y_k \text{ matches format } \langle \text{reasoning} \rangle \ \boxed{\dots}). \quad (14)$$

C Rendering Pipeline

The SVG-only variant converts generated SVG markup to PNG images for the Solver. Process: (1) raw SVG string $\rightarrow$ cairosvg $\rightarrow$ PNG; (2) per-snippet timeout 30 s; (3) parallel workers via ProcessPoolExecutor. Validation: max aspect ratio 100, max dimension 16384; invalid images are discarded. Output: base64-encoded PNG for vLLM/Solver input.

D Training Time

We report the wall-clock training time for MM-Zero on Qwen3-VL-8B-Instruct with the SVG

Table 8: Wall-clock training time for MM-Zero on Qwen3-VL-8B-Instruct (SVG + Filtered) using $8\times$ NVIDIA Pro 6000 (96 GB) GPUs.

Iter	Roles Trained	Data Gen (h)	GRPO (h)	Total (h)
0 $\rightarrow$ 1	Proposer + CodeGen + Solver	15.1	10.0	25.1
1 $\rightarrow$ 2	Proposer + CodeGen + Solver	11.9	8.0	19.9
2 $\rightarrow$ 3	Proposer + CodeGen + Solver	4.5	3.0	7.5
3 $\rightarrow$ 4	Proposer + CodeGen + Solver	4.6	3.0	7.6
4 $\rightarrow$ 5	Proposer + CodeGen + Solver	4.7	3.2	7.9
Total (5 iterations)		40.8	27.2	68.0

rendering pipeline and proposal filtering, trained on $8\times$ NVIDIA Pro 6000 (96 GB) GPUs. Each self-evolving iteration comprises three sequential stages: (1) data generation (Proposer $\rightarrow$ CodeGen $\rightarrow$ SVG rendering $\rightarrow$ filtering), (2) Solver rollout and reward computation, and (3) GRPO training (20 gradient steps per role). Table 8 reports the per-iteration wall-clock time extracted from server logs.

The first two iterations are substantially longer because all three roles (Proposer, CodeGen, Solver) must be trained from the base model for the first time, and the initial data generation produces a larger volume of proposals before filtering stabilizes. From Iteration 2 onward, each cycle takes $\sim 7.5\text{--}8$ h, of which data generation (parallel vLLM inference across 8 GPUs) accounts for roughly 60% and GRPO training accounts for the remaining 40%. The total cost of 5 solver iterations is approximately 68 GPU-hours on Pro A6000s.