RECURSIVE SELF-IMPROVEMENT IN UNIFIED MULTIMODAL MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Unified multimodal models (UMMs) understand and generate both text and images, which lets a model produce its own training data. Existing self-improvement in UMMs keeps supervision on the visual side, where image understanding judges image generation. We propose recursive cross-capability self-improvement (RSI), a training loop in which the text and visual abilities of a UMM supply training data for one another. In each round, the model generates images and reads them to find where it falls short. It then writes programs aimed at these shortcomings, and execution verifies every result against its specification. Verified renders train image generation, while labeled renders and the model's own correct programs train visual understanding and program writing. Program execution thus acts as a source of truth outside the model, so errors do not accumulate across rounds. We study RSI on charts and build BasicChartBench to evaluate open models early in training. On requests worded differently from training, four rounds of RSI raise the score from 45.7% to 60.2%, while continued training stays at 46.3%. Verified construction carries most of the gain, and targeting the model's failures adds 3.5%. Along the way, the share of verified programs rises from 48.9% to 95.2%, and the reader's accuracy on edited renders rises from 55.6% to 87.4%.

1 INTRODUCTION

Progress in foundation models has largely been driven by data that humans collect, filter, and annotate. Unified multimodal models (UMMs) suggest a different path. A UMM understands and generates both text and images within a single model (Chameleon Team, 2024; Zhou et al., 2025; Deng et al., 2025). It could therefore produce its own training data, with one ability supervising another. Self-improvement of this kind has already advanced the reasoning of language models (Zelikman et al., 2022; Yuan et al., 2024). In UMMs, image understanding has recently begun to reward or filter the outputs of image generation (Mao et al., 2025; Jin et al., 2026; Fang et al., 2026). This supervision stays on the visual side, and the text abilities of the model take no part in it.

We ask whether the text and visual abilities of a UMM can improve one another in a closed loop. Image generation shows what the model cannot yet draw. Visual understanding finds what is wrong, and text generation describes what should be drawn instead. Each ability turns what it finds into training data for the others. Round after round, the data follow the model's changing weaknesses (Zhao et al., 2024; Khan et al., 2025), and each gain strengthens the abilities that build the next round. The risk is that the loop feeds on its own outputs. Without an independent check, the errors of one round are learned in the next, and they compound over rounds (Shumailov et al., 2024).

The loop therefore needs a source of truth outside the model. For structured visual content, programs provide one. Infographics, diagrams, tables, and documents can all be written as programs (Yang et al., 2025; Zhuo et al., 2026). Running a program yields an image and an exact record of what it contains, so correctness comes from execution rather than from the model's own judgment. Building on this idea, we propose *recursive cross-capability self-improvement* (RSI). In each round, the model generates images and reads them to find where it falls short. It then writes programs aimed at these shortcomings, and execution verifies every result. Verified renders train image generation. Labeled renders and the model's own correct programs train visual understanding and program writing (Figure 1). The model decides what to learn, and execution ensures that its data are correct.

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

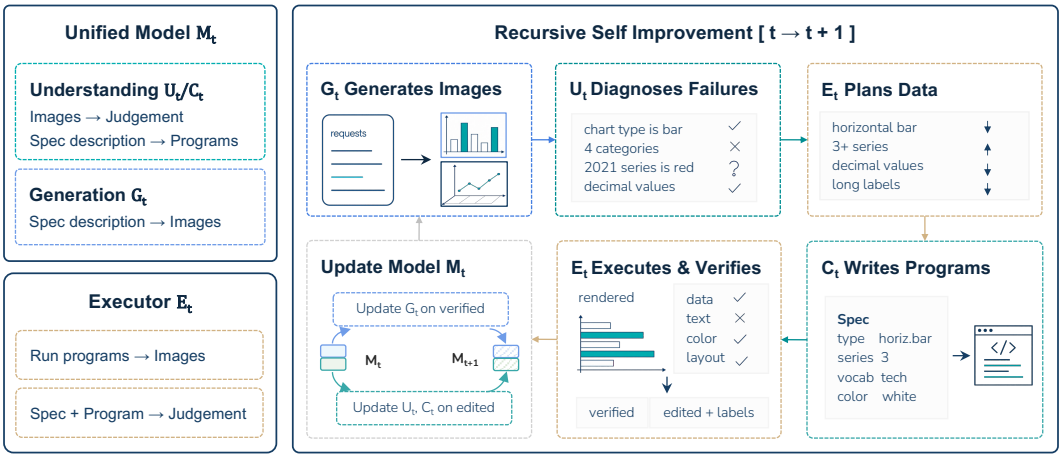


Figure 1: One round of RSI. Executed programs supply verified images and exact labels. The model reads its own generations, and this diagnosis decides what to construct next. The updated model then changes the errors, the reading, and the program writing of the following round.

We study RSI on charts, a representative form of structured visual content. A chart must be exact in its text, values, and structure, and open models remain weak at this. Existing benchmarks are too hard to separate open models at this stage (Zhuo et al., 2026; Li et al., 2026b), so we build BasicChartBench with two levels of difficulty, from familiar content to topics and number formats unseen in training.

Over four rounds, generation improves in every round. On requests worded differently from training, the score rises from 45.7% to 60.2%, while continued training on the original data stays near its starting point. Verified construction carries most of this gain, and building data where the model fails adds to it. The text and visual abilities also improve along the way. After one round, the model writes a correct program for almost every planned example, and its reader catches far more of the errors planted in rendered charts. We close by analyzing in detail where the loop does not yet help.

Our contributions are as follows:

- We introduce the idea of letting the text and visual abilities of a UMM supervise one another in a closed loop, with program execution as an independent check. This check is essential: training on the model's own best-rated images gains only 0.5%, while verified data gain 8.6%.
- We realize this idea as RSI and apply it to charts. To evaluate open models early in training, we build BasicChartBench, with 476 charts and 14,440 atomic questions at two levels of difficulty.
- Over four rounds, RSI raises the score on BasicChartBench L0 from 45.7% to 60.2%, while continued training stays at 46.3%. Targeting the model's failures adds 3.5% over spreading the same data evenly. Along the way, the share of verified programs rises from 48.9% to 95.2%, and the reader's accuracy on held-out edited renders rises from 55.6% to 87.4%.

2 Method

RSI runs in rounds that follow the loop in Figure 1. Section 2.1 sets up notation, and Section 2.2 shows how execution supplies a source of truth. Sections 2.3–2.5 describe the three steps of a round: diagnosing the model's own generations, building verified data, and updating the model.

2.1 Setup

Let M_t denote the UMM after round t, with M_0 the starting model. We use three of its abilities. Image generation G_t produces an image from a text request. Visual understanding U_t answers questions about an image. Program writing C_t, a form of text generation, turns a specification into a program. We call U_t the reader and C_t the writer. The model keeps separate expert parameters for understanding and generation. U_t and C_t share the understanding parameters, and G_t uses the generation parameters. An execution environment $\mathcal{E}$ runs programs and records what they draw.

A specification s fixes the content of one chart. We describe it along predefined capability dimensions, such as chart type, number of categories, number of series, color similarity, value labels, number format, label length, and vocabulary. Most dimensions have ordered levels of difficulty, such as one, two to three, or four to six series. A request $q(s)$ states every requirement of s in natural language, in one of several wordings. Atomic questions derived from s check the requirements one by one. Each question belongs to one of three groups: structure, text, or binding (Section 3).

2.2 Execution as a Source of Truth

For a specification s, the writer produces a program $p = C_t(s)$. Running it gives an image and a record of what was drawn,

$$(I, z) = \mathcal{E}(p). \tag{1}$$

Verification compares the record z with s. It checks the data, text, colors, axes, layout, and text readability of the chart. Accepted pairs $(q(s), I)$ become targets for generation training. Execution thus verifies every target, and the reader only decides what kinds of examples to build.

The same records also supply labels for understanding. We render each verified specification as specified and again after changing one property, such as swapping two colors or reordering categories. Each atomic question is labeled with what the rendered image shows rather than with what the request asks for. If the request asks for a red series and the image draws it in blue, the label marks this requirement as not met. The model's own verified programs, paired with their specifications, train program writing. Together, these examples form the understanding data $\mathcal{D}_t^U$.

2.3 Diagnosing the Model's Own Generations

Each round begins with a fixed set of diagnostic requests. For each request $q(s)$, the current generator produces an image $\widetilde{I} = G_t(q(s))$. This image has no record of what was drawn, so the reader checks its content by answering the atomic questions of s,

$$\widehat{a} = U_t(\widetilde{I}), \qquad d = \mathrm{Compare}(\widehat{a}, s), \tag{2}$$

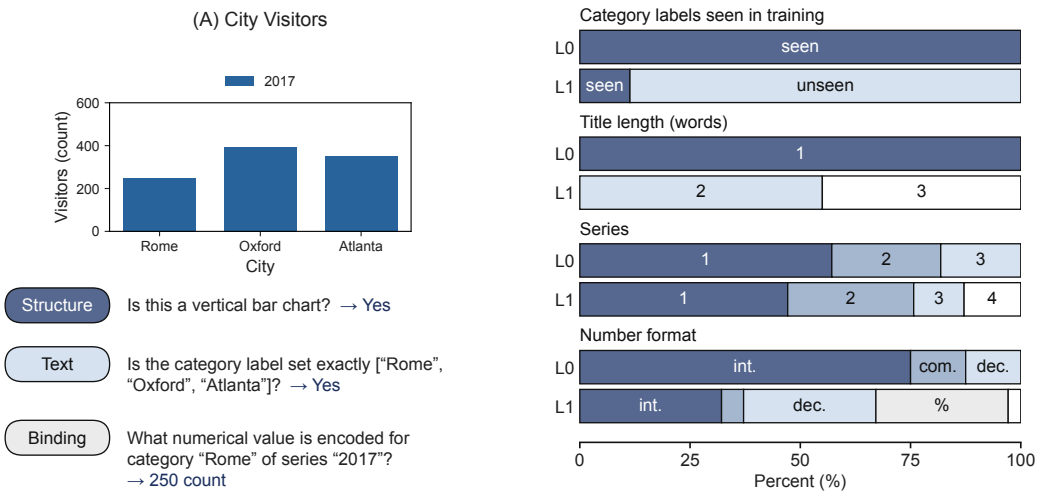
where d marks each requirement as met, violated, or unresolved. The share of requirements met is the generation accuracy. We report it for each capability dimension and each attribute value, such as the size of the values, the number format, the vocabulary, or the wording of the request.

Correct renders of the diagnostic requests calibrate the reader. Every requirement is met on these renders, so each rejection there is a reading error. We add this error rate back to the generation accuracy of the same requirement. Held-out edited renders track in every round whether the reader reports what the image shows or what the request asks. Optical character recognition checks the rendered text independently of the model, and its score decides when a new vocabulary opens.

2.4 From Diagnosis to Verified Data

Fixed rules turn the diagnosis into a data plan, a sampling distribution π_t over specifications. The plan specifies kinds of examples rather than individual charts. It first sets how many specifications to build for each pair of chart type and capability dimension. A fixed 20% of the budget is spread evenly to keep learned skills practiced. The rest follows the diagnosed error rates, with each pair's weight capped at three times the mean. Each dimension of each chart type has a current difficulty level. The level moves up when its generation accuracy reaches 80% and down when it falls below 60%. Within a pair, the dimension being trained uses its current level for 70% of specifications, the next harder level for 20%, and an easier level for 10%. Other dimensions stay at or below their current levels.

The plan then sets attribute values. An attribute value is weak when its generation accuracy is well below the best value of the same attribute and below a mastery level. A weak value may lack training data, or it may be hard despite frequent training, and the two cases call for different data. The rules therefore check how often the value appears in all earlier training data. A value that almost never appears is introduced, and a rare value receives more examples. A common value that is still weak also receives more examples, and the reader receives more examples of it with errors to find. A value that is correct under one request wording and wrong under another is paired with the weaker wording.



(a) Example in BasicChartBench

(b) Distribution of L0 and L1

Figure 2: BasicChartBench. (a) A request, its reference render, and one atomic question per group. (b) How L0 and L1 differ in seen category labels, title length, series count, and number format.

A curriculum widens the space of specifications when fixed gates are met. For example, it adds a number format once accuracy on the current formats reaches 80%. Table 4 lists all thresholds.

The writer produces two candidate programs for each specification drawn from π_t. We keep the first verified candidate and fill each pair up to its share,

$$\mathcal{D}_t^G = \mathrm{Verify}\big(\{(q(s), I) \mid s \sim \pi_t,\ (I, z) = \mathcal{E}(C_t(s))\}\big)\,. \tag{3}$$

Verification also shows where the writer is weak, through its pass rate on each attribute value. Values with a low pass rate receive extra specifications in the next round, up to 20% of the round. Their correct programs are also sampled up to three times as often for the understanding data.

2.5 Updates and Recursion

Both updates start from M_t. The generation update trains only the generation parameters on $\mathcal{D}_t^G$, and the understanding update trains only the understanding parameters on $\mathcal{D}_t^U$. The next model takes the generation parameters from the first update and all other parameters from the second,

$$M_{t+1} = \mathrm{Merge}\big(\mathrm{Learn}_G(M_t; \mathcal{D}_t^G),\ \mathrm{Learn}_U(M_t; \mathcal{D}_t^U)\big). \tag{4}$$

In the next round, G_{t+1} makes new errors, U_{t+1} reads them, and C_{t+1} writes the new examples. The curriculum gates open new parts of the specification space once the diagnosis shows that the earlier parts are learned, so RSI sets its own training space from its diagnosis. At inference time, the generator draws images directly, and programs are used only to build training data.

3 BasicChartBench

Existing chart generation benchmarks are designed to rank strong commercial models (Zhuo et al., 2026; Li et al., 2026b). Early in the training of an open model, most of their requests are too hard, so they separate models poorly. On the chart subsets of StructT2IBench and BizGenEval, for example, our starting model answers only about 11% and 4% of the questions correctly (Section 4.5). We therefore build BasicChartBench, a benchmark for this early stage of training (Figure 2).

Levels. Every chart has one panel. Open models render unfamiliar words poorly, and learning to render arbitrary text needs more training than our budget allows, so the two levels separate familiar text from rare text. Level L0 has 336 charts, 48 for each of the seven chart types, and 10,140 questions. It keeps the vocabulary of the first curriculum stage but uses a request wording that never appears

in training. Level L1 has 140 charts, 20 for each chart type, and 4,300 questions. Its charts come from 14 topics whose words are rare in the generation training data, and they add number formats and category counts beyond the current curriculum. BasicChartBench is never used for diagnosis or data selection, and no request shares its content with a training specification.

Questions. Each request comes with atomic questions, and each question checks one requirement. Structure questions ask about the chart type, the number of categories, the legend, the grid, and the geometry of the marks. Text questions ask whether the title, axis titles, category labels, legend entries, tick labels, and number format appear exactly as requested. Binding questions ask whether values, colors, and order are attached to the right category or series. The diagnostic set and both levels derive their questions and scoring rules from the specification in exactly the same way.

Scoring. A model's score is its generation accuracy, the share of questions whose requirement is met, as judged by the initial reader U_0. U_0 belongs to the starting model and is never updated, so it scores every model by the same fixed criteria. Because U_0 comes from the same model family, we check its trends with an external judge, Qwen3.6-35B-A3B, which shares no weights or training data with our model (Appendix C). We also report the character similarity between the text read by optical character recognition (OCR) and the requested text, which does not depend on our model.

4 Experiments

We ask four questions in turn. Does the loop keep improving image generation (Section 4.2)? Which of its parts drive the gain (Section 4.3)? How do the text and visual abilities of the model change along the way (Section 4.4)? And where does the loop still fall short (Section 4.5)?

4.1 Experimental Setting

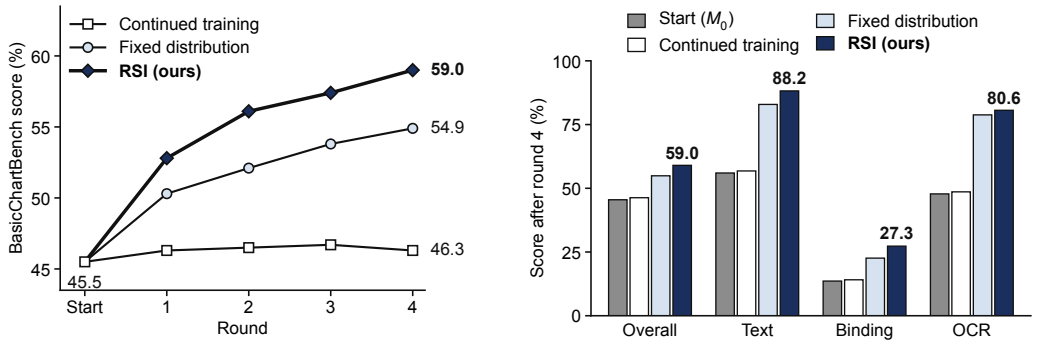
We build on BAGEL-7B-MoT (Deng et al., 2025). The starting model M_0 comes from a curriculum that goes from easy to hard. The model first learns to write plotting programs. Generation then learns basic chart types with short text, followed by a larger set of programmatically generated charts. Finally, the understanding parameters learn to answer atomic questions about program-rendered charts (Appendix A.1). The tasks cover seven chart types that verification can check: vertical bars, horizontal bars, lines, scatter plots, areas, pies, and rings. A diagnostic set of 336 requests in the vocabulary of the first curriculum stage drives the data plan in every round. It varies the series count, value size, and request wording. Every round uses the same training budget (Appendix A).

Controls. Each control removes one part of RSI. *Continued training* trains the generator from M_0 for the same number of steps on the original data mixture of M_0 and tests whether the gain comes only from more training. The *self-filtered* control replaces programs and verification with the reader. It samples two images per request from M_0 and trains on the one that U_0 scores higher. The *fixed distribution* builds the same number of verified charts in each round with the same writer and understanding update as RSI, but it spreads them evenly over pairs of chart type and capability dimension and keeps the first curriculum stage. *Single diagnosis* builds the plan of every round from the diagnosis of M_0 and never diagnoses again. The *frozen reader* diagnoses every round with U_0 instead of the updated reader. The last two share the first round and understanding updates with RSI.

4.2 Does the Loop Keep Improving?

Figure 3 shows generation on BasicChartBench, with its two levels pooled. The score rises in every round, from 45.5% at the start to 52.8%, 56.1%, 57.4%, and 59.0% in the four rounds. Continued training stays near the start (46.3% after round 4), and the fixed distribution reaches 54.9%. The gain holds on both levels, which reach 60.2% on L0 and 56.2% on L1 (Table 1). The external judge shows the same trend, rising from 37.4% to 57.9% on L0 and from 37.5% to 50.0% on L1 (Appendix C). Retraining one round's generator with other seeds moves its score by at most 0.4%.

Most of the gain comes from text and binding. After four rounds, text accuracy is 88.2% against 56.0% at the start, binding is 27.3% against 13.6%, and OCR character similarity is 80.6% against 47.8%. Binding improves partly because text improves. After four rounds, binding accuracy is 34.0% on charts whose category labels are all correct and 25.4% on the others, while at the start the two differ by about 1%. Structure changes much less, as Section 4.5 discusses.



(a) Score after each round

(b) Score after round 4

Figure 3: Generation on BasicChartBench (two levels pooled), scored by the frozen initial reader U_0, which is never updated. (a) Overall generation accuracy after each round for RSI and the two main controls. (b) Overall, text, and binding accuracy and OCR character similarity after round 4.

Table 1: Generation accuracy after each round on the two levels of BasicChartBench, scored by the initial reader U_0. All strategies start from the same model M_0. Single diagnosis and the frozen reader share the first round with RSI (gray). The best strategy in each round is shaded. Rr denotes round r.

Strategy	BasicChartBench L0					BasicChartBench L1				
	Start	R1	R2	R3	R4	Start	R1	R2	R3	R4
Continued training	45.7	46.5	46.5	46.3	46.3	45.1	46.0	46.7	47.8	46.1
Fixed distribution	45.7	51.5	53.8	55.3	56.7	45.1	47.6	47.9	50.3	50.7
Single diagnosis	45.7	54.3	56.9	58.9	59.0	45.1	49.3	50.1	50.9	52.5
Frozen reader	45.7	54.3	56.4	59.1	60.0	45.1	49.3	50.0	51.7	55.2
RSI (ours)	45.7	54.3	57.9	58.9	60.2	45.1	49.3	52.0	54.0	56.2

4.3 Which Parts Drive the Gain?

Execution supplies correct examples. For each of the 15,320 requests of our first round, the self-filtered control keeps the better of two images from M_0 and trains the generator for the same 400 steps. It stays near the start, 0.5% higher on L0 and 1.2% higher on L1. A first-round generator trained on verified charts for the same requests and steps gains 8.6% on L0 and 4.2% on L1. Finding an error does not supply a correct example to learn from, and program execution does.

Diagnosis decides which examples to build. The fixed distribution keeps everything of RSI except the diagnosis. RSI leads it in every round, by 2.8 to 4.1% on L0 and by 1.7 to 5.5% on L1 (Table 1). Choosing examples from the errors in the model's own outputs therefore adds to what verified charts alone provide, and Figure 5 shows one such case in detail. The external judge, which RSI never uses, gives the same ranking as U_0 for every comparison in this section (Figure 4).

Recursion follows the changing model. As the model learns, its errors change, and each understanding update also changes how the reader judges them. Single diagnosis removes the first effect, and the frozen reader removes the second. Both stay close to RSI on L0. On L1, RSI leads single diagnosis by 3.1 and 3.7% in rounds 3 and 4, and it leads the frozen reader by 2.3% in round 3. Diagnosing the changing model thus matters most on the harder level, while updating the reader matters less. Recursion also changes the writer, whose improvement we examine in the next section.

4.4 How Do the Text and Visual Abilities Change?

Program writing and reading share the understanding parameters, so every understanding update changes both of them. Figure 6 tracks both abilities after each round of the loop.

Program writing. With the starting model, 70% of the candidate programs run but only 34% pass verification, so about half of the programs that run draw the wrong content, and 48.9% of

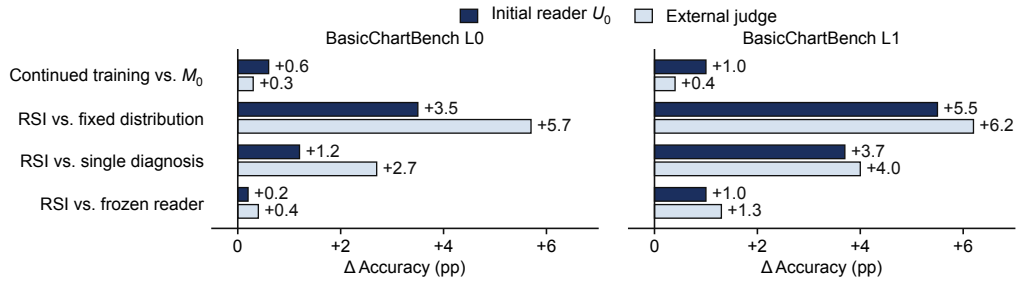


Figure 4: Differences in generation accuracy between RSI and each control on the two levels of BasicChartBench, scored by the initial reader U_0 and by an external judge that RSI never uses. Each row shows the first model minus the second. Comparisons with RSI are after round 4.

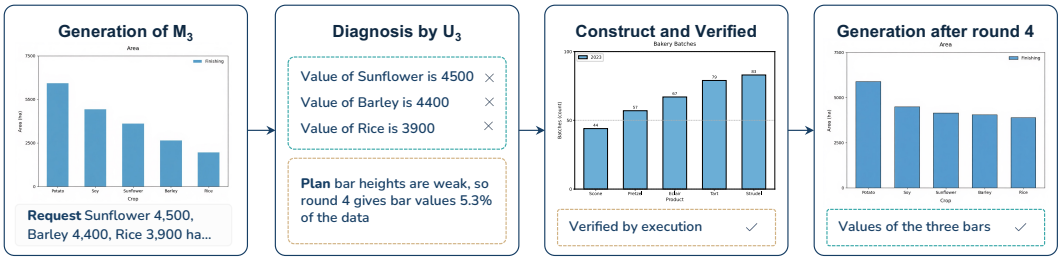


Figure 5: One round of RSI on a diagnostic request. (1) After round 3, the generator draws three bars far too low. (2) The reader U_3 marks exactly these three values as violated, and the plan of round 4 gives this kind of requirement more data. (3) A verified example of the same kind built in round 4. (4) After round 4, the same request is drawn with the three values correct and all text exact.

specifications receive a verified program. After one round the share is 95.2%, and it stays between 95% and 97% in later rounds, including the round that switches to new topics. Horizontal bars rise from 11.8% to 98.4%, and scatter plots from 23.4% to 90.9% (Appendix E). Sampling the writer's correct programs on weak values more often does not explain this gain. When we retrain the first-round understanding update with these programs drawn uniformly and every other example unchanged, the resulting writer passes verification on the same 5,376 second-round specifications at the same rate as ours (90.8% for both). As a side effect, writing the programs of a round takes 18 rather than 159 GPU hours, so each new round of data is much faster to build.

Reading program renders. Correct renders test whether the reader confirms requirements that the image meets. Held-out edited renders come from first-round specifications that no round trains on, so every reader is measured on the same images. On questions whose answer the edit changes, a reader that answers from the request instead of the image is wrong. On correct renders, accuracy rises from 87.1% to 99.3% after four rounds, and binding rises from 77.4% to 98.5%. On changed questions, accuracy rises from 55.6% to 86.5% after one round and then stays near 87%. The reader therefore looks at the image more closely rather than simply accepting more requirements. Building the reading data from the diagnosis brings no measurable benefit, however. A first-round reader trained on questions built from the fixed distribution instead answers 86.3% of the changed questions on the held-out edits of RSI, against 86.5% for our reader, so the two do equally well.

Reading the model's own generations. Program renders cannot show whether the reader also reads the model's own generations, which are the input of every diagnosis. A blind human audit of 392 questions on these generations answers this question (Appendix D). Accuracy rises only from 78.5% for U_0 to 82.4% for U_3 (Table 2), and every reader still accepts about a third of the requirements that humans mark as not shown. The external judge agrees, since the updated readers match it on 83% to 84% of questions, no more than U_0 does. The reader ranks weaknesses well, however. Across pairs of chart type and question group, the Spearman correlation between the violation rates measured by the reader and by humans is 0.82 for U_0 and 0.86 to 0.91 for later readers. The allocation rules

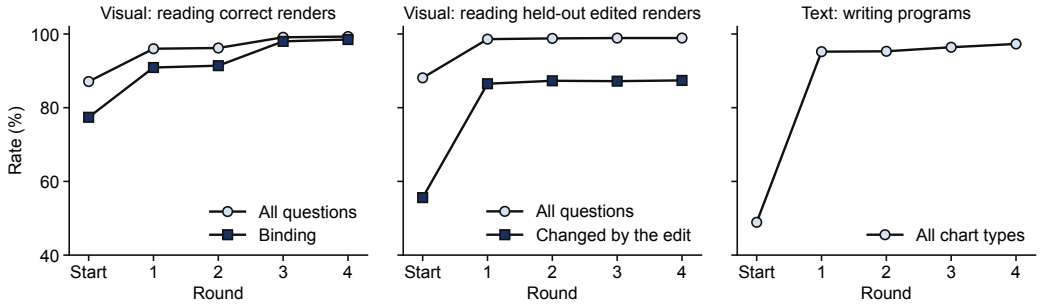


Figure 6: The understanding parameters after each round. Left and middle show reading accuracy on correct renders of the diagnostic set and on held-out edited renders. *Changed by the edit* counts only questions whose answer the edit changes. Right shows the share of specifications for which the model writes a verified program when it builds the data of the next round. Start is the model M_0.

Table 2: Accuracy of each reader on the model's own generations against a blind human audit of the same 392 questions. *False acc.* counts requirements that humans mark as not shown but the reader accepts, and *False rej.* counts the reverse. U_0 gives no parsable answer on 6 questions. The best value in each column is shaded (lowest for the two error rates).

Reader	Structure	Text	Binding	False acc.	False rej.	Questions
U_0 (initial)	83.5	76.7	75.0	34.1	10.6	386
U_1	87.2	85.0	71.4	38.0	2.3	392
U_2	86.5	84.2	75.4	35.8	2.8	392
U_3	85.7	85.7	75.4	34.1	3.8	392

depend on this ranking rather than on exact rates, so the diagnosis points the data to the right places even though many real errors in the model's own images still go unreported by the reader.

4.5 Where Does the Loop Fall Short?

Structure under unseen wording. On the diagnostic set, whose wording matches training, structure accuracy rises from 82.5% to 90.3%. On BasicChartBench, whose wording never appears in training, it first drops and then recovers only partly, reaching 86.1% after round 4. Chart type and panel count still drift. The external judge sees structure recover fully, so part of this drop may come from U_0.

Forgetting when the vocabulary changes. Round 3 replaced the old vocabulary entirely, and exact category text on the old vocabulary fell from 72% to 60% on the diagnostic set. From round 4 on, half of the specifications keep an earlier vocabulary, which brought the old words back in the next round. A curriculum should therefore widen the training data over time rather than replace it.

Existing benchmarks and general abilities. On the chart subset of StructT2IBench (933 charts, 24,925 questions), scored by the external judge with its official questions, RSI raises the score only from 11.0% to 11.9% (Table 3), and continued training reaches 11.7% after 400 steps. On the chart subset of BizGenEval, every model answers about 4% of the questions correctly, and no image passes all of its checks. These benchmarks remain too hard for open models. RSI keeps general generation but costs some general understanding (Table 3). GenEval is 82.9 at the start and between 83.7 and 84.5 after every round, while after four rounds the reader is 2.6% lower on MMBench and 3.0% lower on MMMU. Only 5% of each round's reader data is general visual data.

5 Related Work

Unified Multimodal Models. Unified multimodal models provide a shared foundation for visual understanding and generation, spanning different choices of visual representation, modeling objective, and joint training (Xie et al., 2025; Wu et al., 2025; Deng et al., 2025). Beyond architectural

Table 3: Chart generation on StructT2IBench (chart subset, Qwen3.6-35B-A3B judge) and retained general abilities. GenEval scores the generator, and MME, MMBench, and MMMU the reader.

	Start	Round 1	Round 2	Round 3	Round 4
StructT2IBench	11.0	11.5	11.9	11.7	11.9
GenEval	82.9	83.9	83.7	84.4	84.5
MME	2339	2259	2270	2318	2314
MMBench	82.7	80.1	79.6	79.6	80.1
MMMU	52.1	51.2	50.3	50.3	49.1

unification, interleaved multimodal reasoning connects these capabilities within a task: generated images become intermediate representations that the model can interpret and reason over (Li et al., 2026a; Qin et al., 2026). This makes the interaction between understanding and generation a central aspect of unified modeling. We examine this interaction across training rounds, where understanding the model's own visual outputs informs the construction of its subsequent training data.

Recursive Self-Improvement. Recursive self-improvement couples model updates with the supervision used in later training rounds. Existing approaches recover learning signals from self-generated outputs through outcome verification or learned evaluation (Zelikman et al., 2022; Yuan et al., 2024), and extend this process to adaptive task construction when execution supplies verifiable feedback (Zhao et al., 2025; Heng et al., 2026). For unified multimodal models, the relationship between understanding and generation provides an additional source of supervision through cross-modal consistency and self-generated visual experience (Hong et al., 2026; Thawkar et al., 2026; Fang et al., 2026; Hao et al., 2026). These studies motivate examining how feedback quality evolves alongside the model being trained. Our focus is the role of updated visual understanding in this process: how learning to recognize generation errors changes the examples constructed for subsequent training.

Synthetic Data Generation. The value of synthetic training data depends on both the reliability of its supervision and its relevance to the learner. Programmatic visual data construction exposes information about how examples are produced, supporting annotation and verification through rendering code and executable transformations (Yang et al., 2025; Heng et al., 2026). Adaptive data generation uses learner performance, quality feedback, and coverage gaps to guide the composition of the training set (Khan et al., 2025; Zhao et al., 2024; Yan et al., 2026). These approaches address complementary aspects of data construction: verifying an example does not by itself establish its usefulness for the current model. Our method combines model-based error diagnosis with programmatic construction, using visual understanding to select training needs and execution records to verify the resulting examples and supervise their visible content.

6 Conclusion

We proposed recursive cross-capability self-improvement, a loop in which the text and visual abilities of a UMM produce training data for one another and program execution keeps that data correct. On chart generation, four rounds raise the score on unseen request wording from 45.7% to 60.2%, while continued training on the original data stays near 46%, and an external judge shows the same trend. Verified construction carries the gain, targeting the model's failures adds 3.5%, and program writing and reading improve along with generation. These results point toward a broader goal: models whose abilities keep supplying one another with correct training data as they improve.

AI Use Statement

We used generative AI tools to assist with manuscript formatting and writing polishment. The authors are responsible for verifying all AI-assisted content and take full responsibility for the final manuscript.

References

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint*

arXiv:2405.09818, 2024. URL https://arxiv.org/abs/2405.09818.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025. URL https://arxiv.org/abs/2505.14683.

Zhen Fang, Ruiyan Han, XinYu Sun, Yuchen Ma, Ziheng Wang, Yu Zeng, Zehui Chen, Lin Chen, Wenxuan Huang, Wei-Jie Xu, Yi Cao, and Feng Zhao. UniCorn: Towards self-improving unified multimodal models through self-generated supervision. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13638–13669, 2026. doi: 10.18653/v1/2026.acl-long.621. URL https://aclanthology.org/2026.acl-long.621/.

Ke Hao, Yuanzhi Liang, Tingxi Chen, Rui Li, Haibin Huang, Chi Zhang, Yun Gu, and Xuelong Li. Dreaming in flow: Generative grounding feedback for self-evolving unified multimodal models. *arXiv preprint arXiv:2609.08282*, 2026. URL https://arxiv.org/abs/2609.08282.

Yongrui Heng, Chaoya Jiang, Han Yang, Shikun Zhang, and Wei Ye. EVE: Verifiable self-evolution of MLLMs via executable visual transformations. *arXiv preprint arXiv:2604.18320*, 2026. URL https://arxiv.org/abs/2604.18320.

Jixiang Hong, Yiran Zhang, Guanzhong Wang, Yi Liu, Ji-Rong Wen, and Rui Yan. SUDER: Self-improving unified large multimodal models for understanding and generation with dual self-rewards. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 1686–1697, 2026. doi: 10.1145/3770855.3817654. URL https://arxiv.org/abs/2506.07963.

Weiyang Jin, Yuwei Niu, Jiaqi Liao, Chengqi Duan, Aoxue Li, Shenghua Gao, and Xihui Liu. SRUM: Fine-grained self-rewarding for unified multimodal models. In *European Conference on Computer Vision*, 2026. URL https://arxiv.org/abs/2510.12784.

Zaid Khan, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. DataEnvGym: Data generation agents in teacher environments with student feedback. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.06215.

Ang Li, Charles Wang, Deqing Fu, Kaiyu Yue, Zikui Cai, Wang Bill Zhu, Ollie Liu, Peng Guo, Willie Neiswanger, Furong Huang, Tom Goldstein, and Micah Goldblum. Zebra-CoT: A dataset for interleaved vision-language reasoning. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2507.16746.

Yan Li, Zezi Zeng, Ziwei Zhou, Xin Gao, Muzhao Tian, Yifan Yang, Mingxi Cheng, Qi Dai, Yuqing Yang, Lili Qiu, Zhendong Wang, Zhengyuan Yang, Xue Yang, Lijuan Wang, Ji Li, and Chong Luo. BizGenEval: A systematic benchmark for commercial visual content generation. *arXiv preprint arXiv:2603.25732*, 2026b. URL https://arxiv.org/abs/2603.25732.

Weijia Mao, Zhenheng Yang, and Mike Zheng Shou. UniRL: Self-improving unified multimodal models via supervised and reinforcement learning. *arXiv preprint arXiv:2505.23380*, 2025. URL https://arxiv.org/abs/2505.23380.

Luozheng Qin, Jia Gong, Yuqing Sun, Tianjiao Li, Mengping Yang, Xiaomeng Yang, Chao Qu, Zhiyu Tan, and Hao Li. Uni-CoT: Towards unified chain-of-thought reasoning across text and vision. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.05606.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631:755–759, 2024. doi: 10.1038/s41586-024-07566-y.

Ritesh Thawkar, Shravan Venkatraman, Omkar Thawakar, Abdelrahman Shaker, Fahad Khan, Hisham Cholakkal, Salman Khan, and Rao Muhammad Anwer. Ask, solve, generate: Self-evolving unified multimodal understanding and generation via self-consistency rewards. *arXiv preprint arXiv:2606.27376*, 2026. URL https://arxiv.org/abs/2606.27376.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, and Ping Luo. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12966–12977, 2025. doi: 10.1109/CVPR52734.2025.01210. URL https://arxiv.org/abs/2410.13848.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.12528.

Siyu Yan, Yizhen Gao, Yilin Wang, Dongxing Mao, and Alex Jinpeng Wang. DataEvolver: Self-evolving multi-agent data construction for text-rich image generation. *arXiv preprint arXiv:2606.31537*, 2026. URL https://arxiv.org/abs/2606.31537.

Yue Yang, Ajay Patel, Matt Deitke, Tanmay Gupta, Luca Weihs, Andrew Head, Mark Yatskar, Chris Callison-Burch, Ranjay Krishna, Aniruddha Kembhavi, and Christopher Clark. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17486–17505, 2025. doi: 10.18653/v1/2025.acl-long.855. URL https://aclanthology.org/2025.acl-long.855/.

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

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2203.14465.

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*, 2025. URL https://arxiv.org/abs/2505.03335.

Rui Zhao, Hangjie Yuan, Yujie Wei, Shiwei Zhang, Yuchao Gu, Lingmin Ran, Xiang Wang, Zhangjie Wu, Junhao Zhang, Yingya Zhang, and Mike Zheng Shou. EvolveDirector: Approaching advanced text-to-image generation with large vision-language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2410.07133.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.11039.

Le Zhuo, Songhao Han, Yuandong Pu, Boxiang Qiu, Sayak Paul, Yue Liao, Yihao Liu, Jie Shao, Xi Chen, Si Liu, and Hongsheng Li. Factuality matters: When image generation and editing meet structured visuals. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.05091.

A Implementation Details

A.1 Starting Model

We start from BAGEL-7B-MoT (Deng et al., 2025) and build M_0 with a curriculum that goes from easy to hard. Program writing is first trained on paired ChartNet code and images with the generation parameters frozen. Generation is then trained with the understanding parameters frozen, first on basic chart types with short text and then on a larger set of programmatically generated charts. We use the checkpoint after 14,000 updates of this second stage. Finally, the understanding parameters learn to answer atomic questions about program-rendered charts while keeping program writing, with

the generation parameters frozen. This last step starts from the same code-trained checkpoint and changes only the understanding parameters, so combining it with the generation checkpoint equals running it after generation training. The reader of M_0 is U_0.

A.2 Training in Each Round

Both updates start from the merged model of the previous round and use 64 GPUs. The generation update runs 400 steps with a global batch of 192 on 25,600 unique examples seen three times. It uses AdamW with $\beta = (0.9, 0.95)$, a constant learning rate of 2×10^{-5} after 100 warmup steps, and an exponential moving average with decay 0.995. Verified targets of the round make up 60% of the examples. From the second round on, 10% of these targets are replayed from earlier rounds, sampled evenly across pairs of chart type and capability dimension with double weight on pairs that are already mastered. Replay of the initial generation data makes up 20%, general images 15%, and editing pairs 5%.

The understanding update runs 704 steps with a global batch of 128 on 30,000 examples seen three times. The learning rate decays from 2×10^{-5} to 2×10^{-6} with 3% warmup, and the generation parameters stay frozen. Atomic questions make up 75%, whole-image error reports 5%, code 15%, and general visual pairs 5%. Half of the code examples are the model's own verified programs of the round, weighted by the writer diagnosis. The other half replay the initial code data. Ten percent of the edited-render families are held out in every round to measure the reader.

A.3 Construction and Verification

Each round plans about 36,000 specifications, plus up to 20% more where the writer's success rate is low. Some planned specifications cannot be generated within the retry limit and are dropped, so the rounds use about 35,000 to 43,000. About half of the requests use the curriculum contract wording and the rest use one of six structured forms. The model writes two candidate programs per specification, one greedy and one sampled at temperature 0.6, with at most 6,144 new tokens. A sandbox runs each program and records every panel. Verification checks the data, text, series and category colors, category order, axes and ticks, grid, value labels, legend, layout, and readability. We keep the first verified candidate per specification and fill each pair of chart type and capability dimension up to its share without borrowing from other pairs. If pairs fall short, we build extra specifications for them in proportion to their observed success rate. Only the first round needed this, with about 30,000 extra specifications.

A.4 Decision Rules

Table 4 lists the thresholds. All were fixed before the first round.

B Request Sets and BasicChartBench

The diagnostic set and BasicChartBench L0 have 336 requests each, 48 for each of the seven chart types, and BasicChartBench L1 has 140 requests, 20 for each chart type. In all three sets, the atomic questions are derived from the specification by the same program. Each question checks one requirement and belongs to one of three groups, structure, text, or binding, and all sets use the same scoring rules. No request shares its content fingerprint with any training specification.

Diagnostic set. The diagnostic set uses the vocabulary of the first curriculum stage, which comes from 12 topics of the generation curriculum with one-word titles and axis titles. Every chart has one panel and three to five categories. For the five chart types with axes, the 48 requests cross the number of series (one, two, or three), the kind of series names (years or words), the size of the values (all below 100, or some at least 100), and the request wording (the curriculum contract or one of six structured forms), with two repeats. Area charts need at least two series, so their one-series cells use two. Pies and rings cross value size and wording with twelve repeats. The first repeat uses integers. The second uses decimals for small values and thousands separators for large values. The set has 9,636 questions. After the fourth round, the diagnostic set also gains a part for the second vocabulary, with the same design, 24 requests per chart type, and 4,814 questions. This part only feeds the data plan and the vocabulary gate, and it shares no content with any test set.

Table 4: Decision rules of RSI.

Rule	Setting
Allocation over pairs	20% spread evenly, the rest by violation rate, capped at three times the mean
Mastery of a dimension	Satisfaction of at least 0.80 moves the level up, below 0.60 after an advance moves it down
Weak attribute value	At least 30 resolved questions, at least 15% below the best value of the attribute, and below mastery
Diagnosis by coverage	Below 0.5% of weighted training history is introduced, below 5% is upweighted, otherwise upweighted with extra reading data. Extra share per value is at most 25%
Wording dependence	A gap of at least 15% between wordings pairs the value with its weaker wording
Curriculum gates	Exact category text of at least 70% on the current vocabulary and at least 60% on every earlier vocabulary opens the next vocabulary, exact titles of at least 80% allow two-word titles, and number format satisfaction of at least 0.80 adds the next format
Earlier vocabularies	After a new vocabulary opens, 50% of specifications keep an earlier one
Writer diagnosis	At least 60 specifications and at least 10% below the best value. Extra specifications and code weight grow with the gap, up to three times

BasicChartBench L0. L0 uses the same first-stage vocabulary and a request wording that never appears in training. Every chart has one panel. Half of its charts have three to five categories and half have six to eight, although the curriculum trains only three to five. It has 10,140 questions.

BasicChartBench L1. L1 uses 14 topics whose words appear only rarely in the generation training data, such as finance, transport, healthcare, and music. Every chart has one panel. Titles have two or three words, charts have one to four series, and number formats include decimals and percentages that the curriculum has not yet opened (Figure 2b). It has 4,300 questions.

Question coverage. Structure questions cover the chart type, category count and spacing, legend, grid, axis scale and orientation, and mark geometry. Text questions cover titles, axis titles, category labels, legend entries, tick labels, number format, and text position. Binding questions cover the values encoded by marks, series and category colors, series order, value labels, and simple reasoning over the data.

C Evaluation Protocol and External Judge

Protocol. The judge Qwen3.6-35B-A3B answers each atomic question separately with thinking disabled, and its answers are scored by the same deterministic rules as the reader's. Optical character recognition uses PaddleOCR with rotated reading, so that vertical axis titles are read. For each requested string, we take the best match among the recognized lines. Character similarity is one minus the edit distance to the closest substring, divided by the length of the string.

Agreement with the initial reader. The main results use the frozen initial reader U_0 as the scorer. Because U_0 comes from the same model family and was trained on program-rendered charts, we check its trends against the external judge, which shares no weights or training data with our model. Figure 7 shows both scorers on the two levels of BasicChartBench.

The two scorers agree on the direction of change for the overall score, text, and binding on both levels. On L0, both rise over four rounds on all questions (45.7% to 60.2% for U_0 and 37.4% to 57.9% for the judge), on text (56.8% to 91.7% and 23.9% to 78.6%), and on binding (13.6% to 28.9% and 10.0% to 26.8%). On L1, both rise on all questions (45.1% to 56.2% and 37.5% to 50.0%), on text (54.2% to 80.2% and 23.6% to 45.7%), and on binding (13.6% to 23.6% and 10.3% to 23.3%). They differ on structure. Both see structure fall after the first round, but the judge sees it rise above its starting level by the third round (90.1% to 91.9% on L0 and 90.2% to 93.3% on L1). U_0 stays below its starting level on L0 (87.3% to 85.4% after round 4) and exceeds it on L1 only in round 4 (86.9% to

87.9%). The judge is much stricter on text, so its absolute scores are lower, and we therefore compare trends rather than levels.

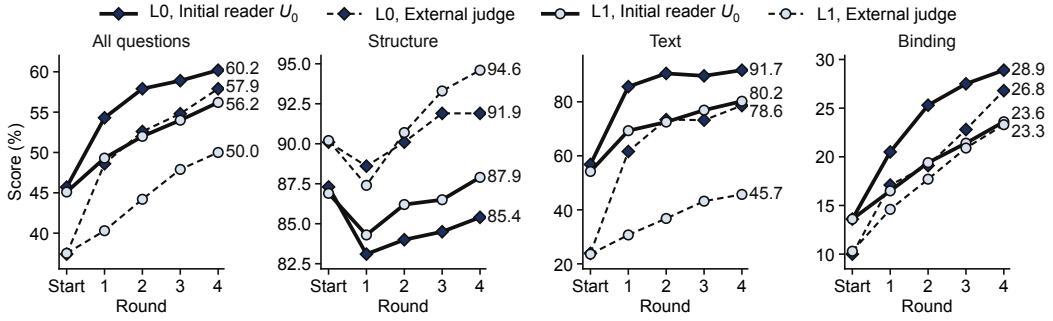


Figure 7: The frozen initial reader U_0 (solid lines) and the external judge Qwen3.6-35B-A3B (dashed lines) on the two levels of BasicChartBench. Scores are satisfaction rates on atomic questions.

D Human Audit of Native Generations

Correct and edited renders (Table 5) come from the same program generator as the reader's training images. They cannot show whether the reader also reads the model's own generations correctly, which is the input of every diagnosis. The native columns of Table 5 answer this question with human labels, and Table 2 breaks the accuracy down by question group.

Separation from RSI. Human labels are collected after the last round and are stored apart from the training pipeline. No human label is used by the diagnosis rules, the construction of specifications, the reader or generation updates, or the thresholds in Table 4. RSI runs identically whether or not the audit exists. The audit only evaluates readers that were already trained.

Images and Questions. The audit uses one shared pool of native images, so that all readers are compared on identical inputs. The pool contains generations of the diagnostic set by M_0 and by the models after rounds 1, 2, and 3, since these are the images the readers U_0 to U_3 had to diagnose. Each model contributes 49 images, 7 per chart type, drawn from disjoint requests, and each image contributes two questions from different question groups. The 196 images give 392 questions, 133 on structure, 133 on text, and 126 on binding. Humans marked 179 requirements as not shown (45.7%) and none as unreadable.

Labeling. Each question is shown as the image and one requirement stated in plain language, and is labeled *shown*, *not shown*, or *unreadable*. Labels are given without knowing which model generated the image, which round it belongs to, or any reader's answer, and the order of items is randomized. Questions labeled unreadable are excluded from accuracy and reported separately.

Metrics. *Acc.* is the share of questions on which the reader's verdict (met or violated) agrees with the human label. *Profile* ρ is the Spearman correlation, across pairs of chart type and question group, between the violation rate measured by the reader and the violation rate measured by humans. It shows whether the reader ranks weaknesses in the same order as humans, which is what the allocation rules use.

Results. Reading of native generations improves less than reading of renders. Accuracy rises from 78.5% for U_0 to 81.4%, 82.1%, and 82.4% for U_1 to U_3. The gain of U_3 over U_0 is 3.9%, with a 95% interval of [0.0, 7.9] from resampling images, so it is at the edge of what 392 questions can resolve. All of the gain comes from fewer false rejections, which fall from 10.6% to 3.8%, mostly on text (76.7% to 85.7%). False acceptances do not fall. Each reader still accepts about a third of the requirements that humans mark as not shown (34.1% for U_0 and U_3, 38.0% for U_1), and accuracy on binding stays near 75%. The reader thus becomes less strict where it was wrongly strict but does not become better at catching real errors in the model's own images, unlike on edited renders (Table 5). The ranking of weaknesses is good from the start and improves a little. The correlation between the violation rates of reader and humans across pairs of chart type and question group is 0.82 for U_0 and

0.86 to 0.91 for U_1 to U_3. Since the allocation rules depend on this ranking rather than on exact rates, the diagnosis points the data to the right places even though about a third of real errors go unreported.

A Pre-Study Check. Before the first round, 210 questions were labeled on 105 native images of M_0 from an earlier development set (seven chart types, three difficulty levels, five images each, and two questions from different groups per image). The same blind protocol was used. Of the 210 requirements, 72 were labeled shown, 138 not shown, and none unreadable. The initial reader agreed with 76.2% of the labels: 88% on structure and binding, and 47% on text. It mostly accepted text that was labeled as not shown. This check led us, before any round, to report optical character recognition alongside the reader for text. The development set is not one of the request sets in Appendix B, and these labels are not part of any reported result.

E Additional Results

Table 5 gives the full reading results behind the left and middle panels of Figure 6, and Table 6 breaks program writing down by chart type.

Table 5: Reading quality of each reader. Correct renders are the reference images of the diagnostic set (9,636 questions). Edited renders are the held-out families of the first round, which no round trains on, so all readers see the same images. *Changed* counts only questions whose answer the edit changes. U_0 is the initial reader, and U_r is the reader after round r. Native columns come from the human audit in Appendix D (392 questions on generations of M_0 to M_3), which covers the readers up to U_3. The best value in each column is shaded.

Reader	Correct renders				Edited renders		Native (reviewed)	
	All	Structure	Text	Binding	All	Changed	Acc.	Profile ρ
U_0 (initial)	87.1	92.3	97.9	77.4	88.1	55.6	78.5	0.82
U_1	96.0	100.0	99.9	90.9	98.6	86.5	81.4	0.86
U_2	96.2	100.0	100.0	91.4	98.8	87.3	82.1	0.91
U_3	99.1	100.0	99.8	98.0	98.9	87.2	82.4	0.89
U_4	99.3	100.0	100.0	98.5	98.9	87.4	–	–

Table 6: Program construction by chart type in each round. Each specification receives two candidate programs. *Executed* is the share of candidates that run and save an image. *Verified* is the share of specifications with at least one candidate that matches the specification. The writer in round r is the model after round $r-1$. Each round uses about 35,000 to 43,000 specifications, and round 1 adds about 30,000 more for pairs that fell short, which are not included. Round 3 uses only the new vocabulary, and round 4 uses the new and the old vocabulary in equal parts.

Chart type	Executed (%)				Verified (%)			
	R1	R2	R3	R4	R1	R2	R3	R4
Vertical bar	73.9	99.1	98.5	97.0	22.5	97.2	98.2	92.0
Horizontal bar	76.2	98.9	99.7	99.8	11.8	86.0	91.7	98.4
Line	83.0	98.3	99.2	99.5	61.6	96.9	95.0	96.3
Scatter	51.6	93.6	95.1	98.0	23.4	89.2	83.7	90.9
Area	88.8	99.7	99.4	99.9	98.5	99.9	99.5	99.7
Pie	82.0	100.0	100.0	100.0	93.3	100.0	99.9	99.9
Ring	19.9	100.0	100.0	100.0	36.3	100.0	99.9	99.8
All	70.5	98.4	98.8	99.1	48.9	95.2	95.3	96.4

F Data Distribution Changes

The rules above were fixed before the round in which they were first used, but some were changed during the study. We list every change that affects the reported rounds, when it was made, and why.

Before round 1. Three fixes were made while the data of round 1 were built, before any training. The pairing of number formats with prompt formats was changed from a fixed probability to a rate that divides the target share by the frequency of the value, because the fixed rate sent 92% of the first-round prompts to one format. The overall share of prompts in the mixed format was then held between 40% and 60%. The content checker was fixed for line charts whose programs plot category names directly, which it had rejected almost always, and the saved traces of round 1 were checked again without running the programs again.

After round 3. Round 3 replaced the old vocabulary entirely, and exact category text on the old vocabulary fell from 72% to 60% on the diagnostic set. From round 4 on, half of the specifications keep an earlier vocabulary once a new one opens. This rule was added because of that observed loss.

After round 4. A diagnostic part for each opened vocabulary and a floor for the old vocabulary in the vocabulary gate were added. They affect only later rounds, which this paper does not report.

G LIMITATIONS

The starting model, the specification space, the capability dimensions, the verification rules, and the curriculum gates all encode prior knowledge. Self-improvement here means that the model produces its own examples and decides where to focus within this structure. The approach applies to domains in which programs can draw the images and the requirements are visible in them. Charts are one such domain, and other structured visual content, such as diagrams, tables, and documents, is a natural next step. Our experiments use one model over four rounds. In this model, the request text passes through the understanding parameters, so training reading and program writing also changes how the generator reads its request. Starting both updates of a round from the same model keeps this effect to about 1%.