CONSISTENCY-DRIVEN CO-EVOLUTION FOR SELF-SUPERVISED CROSS-REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

As chart images, tabular data, and visualization code play increasingly important roles across diverse domains, cross-representation understanding across these modalities poses fundamental challenges for AI systems: the relationships across representations are inherently *one-to-many*, supervision is ambiguous and costly, and model optimization lacks a principled signal that is both direction-adaptive and representation-generalizable beyond task-specific objectives. We introduce **CoCoEvolve** to improve consistency across chart, table, and code representations. Instead of treating cross-representation mapping as a one-to-many problem, we define explicit one-to-one correspondences and optimize models using agreement between representations, without additional annotations. During training, **CoCoEvolve@Train** performs co-evolution across the chart-table-code cycle, while **CoCoEvolve@Test** applies the same consistency objective at inference time for test-time co-optimization. We also present **CoCoEvolve@Eval**, an evaluation suite covering all six cross-representation tasks. Across four benchmarks, **CoCoEvolve** improves performance in both training-time and test-time settings.

1 INTRODUCTION

Chart images are ubiquitous across scientific publications (Wang et al., 2024b; Yang et al., 2025b; Guo et al., 2026), financial reports (Shu et al., 2025), data analyses (Huang et al., 2024), etc. The structured information they convey can be expressed in multiple forms (Zhao et al., 2025; Tang et al., 2025; Zhao et al., 2025): a *chart image* encodes data visually through axes, marks, and layouts; a *data table* represents the core information in structured tabular form; and *visualization code* captures the declarative logic that transforms data into visual output. Understanding and reasoning across different domains and representations poses great challenges for AI systems, as it demands not merely visual perception but precise *structural cross-modal understanding*: the ability to reason about data semantics, encode logic, and establish representational correspondence across modalities.

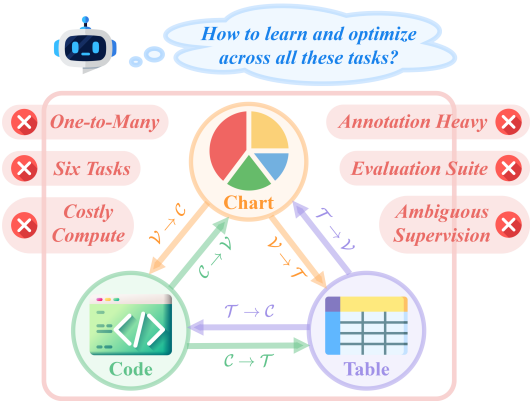


Figure 1: **Chart-Table-Code Representation Cycle.** Cross-representation understanding across six tasks poses great challenges for model learning and optimization due to its inherent one-to-many nature, supervision ambiguity, and high computational cost (§1).

A central bottleneck in cross-representation understanding is the **ambiguity and cost of supervision**. The relationships across chart images, tabular data, and rendering code are inherently *one-to-many*: a single chart image may correspond to multiple valid tabular representations, and equally, multiple valid rendering programs. This means each *chart-table-code* instance requires annotating a large space of potential correspondences, making labeled supervision not only expensive to produce at scale, but fundamentally ill-addressed by existing benchmarks (Kondic et al., 2026) that *unconstrained one-to-one* ground truths (§B.1). A **principled constraint definition** is needed to ground such *one-to-many* mappings to precise *one-to-one* correspondences (§3.1). Chart images are abundant, yet *accurately labeled cross-representation correspondences* remain scarce and poorly defined.

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.

Compounding this, even with principled constraint definitions in place, it remains challenging to **establish a principled optimization signal** that is both direction-adaptive and representation-generalizable. A model trained on fixed chart-to-table or chart-to-code pairs can exhibit degraded performance not only on the trained tasks (*chart-to-table* & *chart-to-code*), but also on tasks in the reversed direction (*e.g., table-to-chart*) or with unseen representation combinations (*e.g., table-to-code*) (§B.2). Such task-specific objectives operate locally on fixed representation pairs and cannot enforce global semantic correctness or generalize to unseen directions and representation combinations. This calls for a principled supervision paradigm: *one that operates agnostically across tasks, directions, and representations, enforcing global semantic correctness without relying on fixed ground-truth labels*.

Key Insight

With a principled constraint definition (§3.1), *chart*, *table*, and *visualization code* are placed on equal footing as *multiple representations of the same underlying semantics*. If each representation is correct, they should mutually agree — this agreement can serve as a *principled optimization signal* requiring no annotation and remaining agnostic to tasks, directions, and representations (§B.3).

Building on these insights, we introduce **CoCoEvolve** (§3), a consistency-driven co-evolve framework that jointly addresses both challenges. By proposing a *principled constraint definition* (§3.1), **CoCoEvolve** unambiguously grounds *one-to-many* mappings to precise *one-to-one* correspondences. Also, rather than relying on labeled correspondences and task-specific learning, **CoCoEvolve** leverages cross-representation agreement as a *principled, annotation-free optimization signal* that is agnostic to tasks, directions, and representations, training models through a *representation cycle* that enforces global semantic correctness at scale. To sum up, our main contributions are:

1. We introduce a *principled constraint definition* (§3.1) that explicitly addresses the inherent *one-to-many* ambiguity overlooked by existing benchmarks and methods.
2. We present **CoCoEvolve** (§3), a co-evolution framework that turns *cross-representation consistency* into a *principled self-supervision signal*, enabling annotation-free optimization at both train (§3.2) and test time (§3.3).
3. We propose a systematic and unified evaluation suite (§3.5) for all tasks in the cycle (§3.1), addressing the key limitations of existing LLM- and MLLM-based evaluation approaches.
4. Experiments demonstrate that **CoCoEvolve** effectively enhances model cross-representation understanding abilities, yielding up to $\uparrow 37.91\%$ gains on the non-overlapping test set, and is generalizable to *out-of-domain* settings with improvements of up to $\uparrow 46.88\%$ (§4).

2 Related Work

Cross-Representation Learning. Recent work shows promising progress on individual edges of the *chart-table-code* representation cycle (§3.1). *Chart-to-table extraction* focuses on recovering structured tabular data from chart images (Meng et al., 2024; Liu et al., 2023; Kondic et al., 2026), while *chart-to-code generation* targets the visualization program underlying a chart (Yang et al., 2025b; Tang et al., 2025; Zhao et al., 2025; Kondic et al., 2026), leveraging code-capable multimodal models to reproduce the rendering logic. However, these lines of work share a common limitation: *they treat each edge as an independent supervised task, requiring costly labeled correspondences and ignoring the natural semantic redundancy across three representations*. This reveals the critical cross-representation learning gap that our work aims to bridge via co-evolution.

Self-Supervised Cycle Learning. Self-supervised learning shows its strengths in annotation-free learning across various domains (Chen & Krishnan, 2021; Fang & Xie, 2022; Li et al., 2024). Cycle consistency, as a self-supervision signal, is introduced in the vision domain by CycleGAN (Zhu et al., 2017), revealing that unpaired cross-domain translation can be learned by enforcing round-trip reconstruction. This principle has since been extended to language (Shen et al., 2025) and vision settings (Adrian et al., 2024). However, *existing approaches either rely on single representations that are poorly suited for structured cross-modal reasoning or are limited to specific domains*. This motivates our consistency-driven co-evolution framework that unifies cycle consistency and co-training into a single annotation-free paradigm for multimodal cross-representation learning.

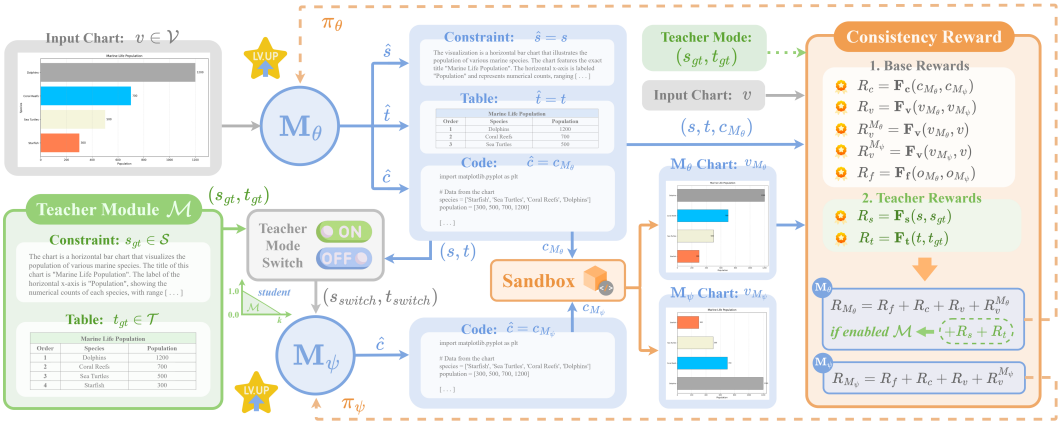


Figure 2: **CoCoEvolve Overview.** We introduce the consistency-driven co-evolution framework. Dashed arrows (- - -) denote gradient updates at training time; models are frozen at test time.

3 METHOD

3.1 CROSS-REPRESENTATION LEARNING

Problem Formulation. Let $\mathcal{V}$, $\mathcal{T}$, and $\mathcal{C}$ denote the spaces of chart images, tabular data, and visualization code, respectively. We study the problem of cross-representation understanding as *six* tasks in a *representation cycle* across these three spaces: $\mathcal{V} \Leftrightarrow \mathcal{T}$, $\mathcal{T} \Leftrightarrow \mathcal{C}$, and $\mathcal{V} \Leftrightarrow \mathcal{C}$.

Representation Cycle. We define three directional mappings that together constitute the chart-table-code cycle:

$$f_\theta : \mathcal{V} \to \mathcal{T}, \quad g_\psi : \mathcal{T} \to \mathcal{C}, \quad h : \mathcal{C} \to \mathcal{V} \tag{1}$$

where f_θ performs *chart-to-table* decoding of a chart image $v \in \mathcal{V}$ into a tabular representation $t \in \mathcal{T}$, g_ψ performs *table-to-code* generation from tabular data $t \in \mathcal{T}$ to visualization sandbox $c \in \mathcal{C}$, and h denotes a deterministic code executor that renders c back into a chart image. Together, these form a full cycle $h \circ g_\psi \circ f_\theta : \mathcal{V} \to \mathcal{T} \to \mathcal{C} \to \mathcal{V}$.

Principled Constraint Definition. Existing work (Kondic et al., 2026) fails to reasonably account for the inherent one-to-many nature of cross-representation mappings (§1). To mitigate this ambiguity, we augment g_ψ with auxiliary constraints $s \in \mathcal{S}$, where $\mathcal{S}$ denotes the space of descriptive constraints, and each s_i conditions the alignment of t_i and c_i with respect to v_i.

$$g_\psi(\cdot \mid s) : \mathcal{T} \to \mathcal{C} \tag{2}$$

where $g_\psi(\cdot \mid s)$ represents a family of conditioned *table-to-code* mappings $g_\psi(\cdot \mid s) : s \in \mathcal{S}$.

3.2 COCOEVOLVE@TRAIN: TRAIN-TIME CONSISTENCY-DRIVEN CO-EVOLUTION

By sharing a *chart-table-code representation cycle* (§3.1), **CoCoEvolve** enables the co-evolution of two models, M_θ and M_ψ, jointly over unannotated chart images $\mathcal{V} = \{v\}$, where π_θ and π_ψ (Eq.8) denote the respective policies parameterizing $f_\theta : \mathcal{V} \to \mathcal{T}$ (Eq.1) and $g_\psi : \mathcal{T} \to \mathcal{C}$ (Eq.2). Leveraging hierarchical consistency-driven training signals below, **CoCoEvolve** proposes a novel co-evolution objective (Eq.8) that serves as a drop-in training signal for existing RL algorithms, such as GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), and GSPO (Zheng et al., 2025).

Code Consistency Reward. $\mathbf{F}_c$ measures semantic agreement between $\hat{c}_\theta$ and $\hat{c}_\psi$ through execution success and embedding similarity:

$$\mathbf{F}_c^{(\pi)}(\hat{c}_\theta, \hat{c}_\psi) = \omega_e^{(\pi)} \cdot \mathbb{1}[h(\hat{c}_\theta) \wedge h(\hat{c}_\psi)] + \omega_s^{(\pi)} \cdot \mathtt{sim}_\mathrm{c}(\hat{c}_\theta, \hat{c}_\psi) \tag{3}$$

where $\pi \in \{\theta, \psi\}$ indexes the model, and $\omega_e^{(\pi)}, \omega_s^{(\pi)}$ are per-model sub-weights. where $\mathtt{sim}_\mathrm{c}$ is a normalized cosine similarity:

$$\mathtt{sim}_\mathrm{c}(c_a, c_b) = \left[\frac{\cos(e_a, e_b) - \cos(e_a, e_{base})}{1 - \cos(e_a, e_{base})} \right]_0^1 \tag{4}$$

where e_a and e_b are embeddings of c_a and c_b, e_{base} is the baseline embedding anchoring similarity above generic code patterns, and $[\cdot]_0^1$ denotes clamping to $[0, 1]$.

Visual Consistency Reward. $\mathbf{F}_v$ measures visual agreement between two rendered chart images through four complementary metrics, gated on code execution success:

$$\mathbf{F}_v^{(\pi)}(v_a, v_b) = \sum_m \omega_m^{(\pi)} \cdot F_m(v_a, v_b) \tag{5}$$

where $m \in \{clip, ssim, ocr, dino\}$, and $F_m \in \{F_{clip}, F_{dino}, F_{ssim}, F_{ocr}\}$ (Eq.20,21,22,23) covers four complementary dimensions of chart evaluation (§F.1.1), respectively.

Format Reward. $\mathbf{F}_f$ enforces well-formed structured outputs as a binary reward:

$$\mathbf{F}_f(c, \pi) = \mathbb{1}\left[\texttt{format}_\pi(c)\right] \tag{6}$$

where $\texttt{format}_\pi(c)$ is a model-specific boolean predicate that verifies if c conforms to the output format required by model $\pi \in \{\pi_\theta, \pi_\psi\}$.

Hierarchical Reward. The full hierarchical reward for each model aggregates the above components:

$$R_\pi = \lambda_f^{(\pi)} \mathbf{F}_f(\hat{c}_\pi) + \lambda_c^{(\pi)} \mathbf{F}_c^{(\pi)}(\hat{c}_\theta, \hat{c}_\psi) + \lambda_\pi \mathbf{F}_v^{(\pi)}(\hat{v}_\pi, v) + \lambda_v^{(\pi)} \mathbf{F}_v^{(\pi)}(\hat{v}_\theta, \hat{v}_\psi) \tag{7}$$

where $\pi \in \{\theta, \psi\}$, and $\hat{v}_\pi = h(\hat{c}_\pi)$ denotes the rendered chart image of model π via h (Eq.1). The shared reward terms $\mathbf{F}_c$ (Eq.3) and $\mathbf{F}_v(\hat{v}_\theta, \hat{v}_\psi)$ (Eq.5) enforce mutual agreement between two models, while $\mathbf{F}_v(\hat{v}_\theta, v)$ and $\mathbf{F}_v(\hat{v}_\psi, v)$ (Eq.5) independently ground each model against the original v, preventing degenerate solutions where both models collude to produce mutually consistent but semantically incorrect outputs.

Co-Evolve Training Objective. Unlike independent multi-task learning, π_θ and π_ψ are coupled through both sampling and reward: π_ψ conditions on π_θ's output $(\hat{s}^{(i)}, \hat{t}^{(i)})$, and each model's advantage is derived from a reward functional that depends on the other model's output. For each chart image $v \in \mathcal{V}$, π_θ draws K_θ rollouts $o_\theta^{(i)} \sim \pi_{\theta,\text{old}}(\cdot \mid v)$; and for each rollout $o_\theta^{(i)} = (\hat{s}^{(i)}, \hat{t}^{(i)}, \hat{c}_\theta^{(i)})$, π_ψ draws K_ψ conditioned rollouts $o_\psi^{(i,k)} \sim \pi_{\psi,\text{old}}(\cdot \mid \hat{s}^{(i)}, \hat{t}^{(i)})$. The joint objective is (§D):

$$\mathcal{J}(\theta, \psi) = \mathbb{E}_v \, \mathbb{E}_{\substack{o_\theta^{(i)} \sim \pi_{\theta,\text{old}}(\cdot|v) \\ o_\psi^{(i,k)} \sim \pi_{\psi,\text{old}}(\cdot|\hat{s}^{(i)}, \hat{t}^{(i)})}} \Bigg[\underbrace{\frac{1}{K_\theta} \sum_{i=1}^{K_\theta} \mathcal{L}_\theta^{(i)}}_{\pi_\theta \text{ surrogate}} + \underbrace{\frac{1}{K_\theta K_\psi} \sum_{i=1}^{K_\theta} \sum_{k=1}^{K_\psi} \mathcal{L}_\psi^{(i,k)}}_{\pi_\psi \text{ surrogate}} \Bigg] \tag{8}$$

3.3 CoCoEvolve@Test: Test-Time Consistency-Driven Co-Optimization

Mirroring co-training, **CoCoEvolve** further optimizes M_θ and M_ψ at test time. Given an input chart image v, **CoCoEvolve** produces the best outputs from both models:

$$c_\psi^{(k^*)} = \underset{k \in [K_\psi]}{\arg\max} \, R_\psi \left(\hat{c}_\psi^{(k)} \mid \underset{i \in [K_\theta]}{\arg\max} \, R_\theta \left(o_\theta^{(i)} \mid v \right) \right) \tag{9}$$

where $o_\theta^{(i)} = (\hat{s}^{(i)}, \hat{t}^{(i)}, \hat{c}_\theta^{(i)}) \sim f_\theta(\cdot \mid v)$ are K_θ candidate rollouts from f_θ, $c_\psi^{(k)} \sim g_\psi(\cdot \mid \hat{s}^{(i^*)}, \hat{t}^{(i^*)})$ are K_ψ candidate rollouts from g_ψ conditioned on the best f_θ output $o_\theta^{(i^*)}$. R_θ and R_ψ (Eq.7) serve as annotation-free signals for test-time optimization. The final outputs $(\hat{t}^{(i^*)}, \hat{c}_\theta^{(i^*)}, \hat{c}_\psi^{(k^*)})$ cover all three representations of the cycle across six tasks (§3.1).

3.4 Seamless Extension to Supervised Consistency-Driven Co-Learning

When high-quality annotations are available, **CoCoEvolve** is also seamlessly extensible to supervised settings at both train and test time via an optional **teacher module** $\mathcal{M}$.

Teacher-Guided Grounding. For a fraction τ of samples per batch, $\mathcal{M}$ replaces π_θ's outputs $(\hat{s}^{(i)}, \hat{t}^{(i)})$ with ground-truth (s_{gt}, t_{gt}) as π_ψ's conditioning context, decoupling π_ψ from π_θ's potentially corrupted outputs on those samples. Formally, for each sample, the conditioning context is:

$$(\hat{s}_\psi^{(i)}, \hat{t}_\psi^{(i)}) = \begin{cases} (s_{gt}, t_{gt}) & p = \alpha \\ (\hat{s}^{(i)}, \hat{t}^{(i)}) & p = 1 - \alpha \end{cases} \tag{10}$$

where p denotes sampling probability. In practice, we implement $\mathcal{M}$ via *stratified sampling with randomized rounding* where exactly $\lfloor \alpha B \rfloor$ or $\lfloor \alpha B \rfloor + 1$ samples per batch of size B are teacher-guided, ensuring the minimum-variance unbiased estimator of the target fraction α under a fixed batch size.

Teacher Reward. When $\mathcal{M}$ is enabled, π_θ receives two additional grounding rewards:

$$R_s = \mathbf{F}_s^{(\theta)}(\hat{s}, s_{gt}) = \cos(\phi(\hat{s}), \phi(s_{gt})) \tag{11}$$

$$R_t = \mathbf{F}_t^{(\theta)}(\hat{t}, t_{gt}) = \sum_m \omega_m^{(\theta)} \cdot F_m(\hat{t}, t_{gt}) \tag{12}$$

where $m \in \{schema, value\}$, $\phi(\cdot)$ is a sentence embedding function via SentenceBERT (Reimers & Gurevych, 2019), and $F_m \in \{F_{schema}, F_{value}\}$ denote column-level schema F1 and cell-level value F1 over all $(row, column, value)$ triples, respectively. The full π_θ reward under *teacher* mode extends from Eq.7 to become:

$$R_\theta^+ = \lambda_f^{(\theta)} \mathbf{F}_f(\hat{c}_\theta) + \lambda_c^{(\theta)} \mathbf{F}_c^{(\theta)}(\hat{c}_\theta, \hat{c}_\psi) + \lambda_\theta \mathbf{F}_v^{(\theta)}(\hat{v}_\theta, v) + \lambda_v^{(\theta)} \mathbf{F}_v^{(\theta)}(\hat{v}_\theta, \hat{v}_\psi) + \lambda_s^{(\theta)} R_s + \lambda_t^{(\theta)} R_t \tag{13}$$

Linear Annealing. For robust co-evolution, **CoCoEvolve@Train** incorporate a teacher-guidance fraction α linearly annealed from α_{start} to α_{end} over N_{train} training steps:

$$\alpha(t) = \alpha_{start} + \frac{t}{N_{train}} (\alpha_{end} - \alpha_{start}) \tag{14}$$

where t denotes current training step. This allows **CoCoEvolve@Train** to transition gradually from teacher-guided co-training toward fully self-supervised co-evolution, as π_θ becomes increasingly reliable over the course of training.

3.5 CoCoEvolve@Evaluation Suite for Multidimensional Assessment

In addition to the lack of a systematic and generalizable evaluation suite, existing LLM-as-Judge and MLLM-as-Judge approaches for chart-to-code assessment suffer from ill-grained judgment, typically relying on LLMs and MLLMs to generate the final score by collapsing complex visual and structural fidelity into a single 0-100 score (Tang et al., 2025; Yang et al., 2025b), a scale at which LLMs and MLLMs are known to be inconsistent and unreliable (Wang et al., 2024a; Stureborg et al., 2024). This makes them ill-suited for the accurate, multidimensional assessment that charts and code demand. To address both limitations, we introduce **CoCoEvolve@Eval**, a unified evaluation suite that, for the first time, systematically covers all six cross-representation tasks in the cycle. Rather than collapsing assessment into wide-range scores or delegating final scoring to LLMs and MLLMs, our evaluation suite decomposes evaluation into interpretable dimensions to obtain multidimensional means. As such, **CoCoEvolve@Eval** yields multidimensional measurements generalizable across benchmarks and evaluation scenarios.

Rule-as-Judge. Our rule-as-jugde for chart, code, table, and constraint evaluation leverages deterministic rubric metrics, covering structural, semantic, perceptual, textual, and stylistic accuracy for charts (F_{chart}, §F.1.1); executability, quality, structural, lexical, contextual, and semantic accuracy for code (F_{code}, §F.1.2); schema and value accuracy for tables (F_{table}, §F.1.3); and semantic and lexical accuracy for constraints ($F_{constraint}$, §F.1.4). Full metric definitions are provided in §F.1.

LLM-as-Judge. To complement rule-based metrics with semantic reasoning beyond surface-level code similarity, we introduce an LLM-as-judge evaluator that assesses visualization code quality across five fine-grained dimensions: data correctness, chart type accuracy, structural fidelity, visual accuracy, and style accuracy (§F.2). Rather than collapsing each assessment into a wide-range score where LLMs are known to be unreliable (§3.5), each dimension is scored independently and weighted into a final score J_{code}.

MLLM-as-Judge. To directly assess visual fidelity between predicted and reference outputs, we introduce MLLM-as-judge for charts, tables, and constraints. Each representation is evaluated across five fine-grained dimensions targeting distinct aspects of prediction quality, weighted into final scores J_{chart} (§F.3.1), J_{table} (§F.3.2), and $J_{constraint}$ (§F.3.3).

Table 1: **Performance Evaluation on Code.** We evaluate $\mathcal{X} \to \mathcal{C}$ ($\mathcal{X} \in \{\mathcal{V}, \mathcal{T}\}$) performance for M_θ ($\mathcal{V} \to \mathcal{C}$) and M_ψ ($\mathcal{T} \to \mathcal{C}$) using *rule-as-judge* and *LLM-as-judge*. We calculate the mean score over four rollouts for both M_θ and M_ψ, denoted as *m@4*. Compared to *baselines*, **CoCoEvolve**-enhanced models are optimized at train time and/or test time in *student* mode.

Model	Exe. (%)	Rule-as-Judge (%)						LLM-as-Judge (%)					
	m@4	bleu	str.	lex.	ctx.	sem.	m@4	type	data	str.	visual	style	m@4
Baselines													
Qwen3-VL-2B	58.56	29.92	32.62	34.39	92.99	76.12	53.21	55.37	24.26	30.79	16.16	24.03	30.12
Qwen3-VL-4B	78.47	30.52	33.84	37.14	94.02	77.88	54.68	74.35	34.03	39.31	23.94	33.38	41.00
InternVL3.5-4B	59.72	27.91	35.99	32.02	92.29	75.28	52.70	58.10	28.33	33.61	18.89	23.75	32.54
Llama3.2-3B	22.51	29.15	10.79	21.50	84.32	61.97	41.55	43.69	30.42	23.08	13.60	19.05	25.97
DeepSeek-1.3B	47.69	26.89	27.89	26.44	91.97	72.05	49.05	45.73	29.51	25.08	16.49	16.47	26.66
DeepSeek-6.7B	68.00	28.71	29.81	30.54	92.26	74.91	51.25	59.27	34.93	30.34	21.05	24.65	34.05
Qwen3-1.7B	57.47	27.99	30.03	31.90	93.32	77.86	52.22	63.78	33.63	32.56	19.26	26.56	35.16
Qwen3-4B	77.60	31.56	26.08	33.02	91.94	71.72	50.86	76.66	35.75	34.31	23.03	28.08	39.57
Qwen3-8B	82.75	30.32	29.89	32.19	93.65	75.28	52.27	80.83	37.95	37.73	26.34	32.52	43.07
CoCoEvolve (@Test)													
Qwen3-VL-2B	87.04	31.54	38.36	37.78	93.22	77.37	55.65	76.85	35.56	41.67	24.44	35.00	42.70
Qwen3-VL-4B	93.52	33.09	41.21	41.91	96.27	81.18	58.73	86.30	44.26	48.70	34.81	44.81	51.78
InternVL3.5-4B	92.59	31.78	45.87	38.23	95.86	80.67	58.48	75.19	41.30	44.81	28.33	34.63	44.85
Llama3.2-3B	85.19	34.87	29.95	32.06	94.73	76.70	53.66	59.26	48.52	37.78	28.15	29.07	40.56
DeepSeek-1.3B	96.30	32.23	42.37	34.08	96.47	81.01	57.23	69.44	46.33	44.63	34.63	29.81	44.97
DeepSeek-6.7B	99.07	34.23	45.48	40.81	97.50	83.18	60.24	88.33	48.89	50.19	41.48	42.96	54.37
Qwen3-1.7B	97.22	33.32	35.34	38.55	92.65	74.09	54.79	88.52	48.67	45.07	39.44	42.52	52.84
Qwen3-4B	99.07	34.94	36.87	39.50	95.93	79.64	57.38	93.33	50.74	48.89	41.85	44.26	55.81
Qwen3-8B	100.00	35.56	44.92	41.70	97.43	83.04	60.53	93.89	52.96	53.33	46.11	49.44	59.15
CoCoEvolve (@Train)													
Qwen3-VL-2B	85.42	30.84	34.58	36.79	94.73	79.37	55.26	57.68	27.37	33.24	18.79	26.39	32.69
Qwen3-VL-4B	92.59	31.36	35.06	39.91	95.22	80.92	56.49	75.70	36.15	40.80	26.11	35.28	42.81
Llama3.2-3B	83.10	29.32	29.56	32.53	94.30	78.74	52.89	42.80	32.40	27.16	16.60	20.83	27.96
DeepSeek-1.3B	93.00	27.75	34.65	29.39	92.70	76.77	52.25	49.99	32.73	33.43	21.74	17.24	31.03
Qwen3-1.7B	76.68	29.02	32.28	34.93	94.98	79.38	54.12	64.83	36.01	33.54	22.56	25.20	36.43
Qwen3-4B	95.25	32.80	33.47	41.32	96.37	80.41	56.87	77.86	37.16	35.72	25.40	30.59	41.35
Qwen3-8B	98.15	33.89	36.02	42.19	97.65	82.28	58.41	83.28	39.24	39.12	28.74	34.27	44.93
CoCoEvolve (@Train + @Test)													
Qwen3-VL-2B	93.36	33.90	40.44	39.14	94.76	79.42	57.53	79.32	37.37	42.83	25.69	35.88	44.22
Qwen3-VL-4B	97.22	36.31	41.36	43.87	96.26	82.32	60.02	87.96	46.48	51.11	38.34	43.33	53.44
Llama3.2-3B	100.00	36.02	41.34	38.31	97.19	82.30	59.03	78.70	50.74	42.41	34.63	37.59	48.81
DeepSeek-1.3B	99.07	33.08	44.93	37.48	97.28	81.43	58.84	80.19	48.48	46.85	35.37	35.93	49.36
Qwen3-1.7B	100.00	34.32	49.47	43.75	97.72	84.52	61.96	88.85	50.74	49.81	39.63	42.22	54.25
Qwen3-4B	100.00	36.47	42.73	41.57	97.42	84.03	60.44	93.48	51.33	49.85	42.81	45.33	56.56
Qwen3-8B	100.00	39.65	50.37	43.68	98.88	88.23	64.16	94.62	54.77	55.38	47.52	50.79	60.62

4 EXPERIMENTS

4.1 SETUP

Data. We randomly sample non-overlapping training and test subsets from *ChartCoder* (Zhao et al., 2025) through outcome-grounded filtering (§C). For main evaluation, we measure model performance on six cyclic tasks of the *chart-table-code* learning cycle (§3.1), including $\mathcal{V} \Leftrightarrow \mathcal{T}$, $\mathcal{V} \Leftrightarrow \mathcal{C}$, and $\mathcal{C} \Leftrightarrow \mathcal{T}$. To evaluate model generalizability and robustness, we extend our evaluation to (1) *out-of-domain data*: multi-domain test sets adapted from *ChartMimic* (Yang et al., 2025b) (*e.g., physics, mathematics, economics, biology, etc.*) and *ChartNet* (Kondic et al., 2026) (*e.g., health, finance, etc.*);

Table 2: **Performance Evaluation on Chart.** We evaluate $\mathcal{C} \rightarrow \mathcal{V}$ performance for M_θ ($\mathcal{V} \rightarrow \mathcal{C}$) and M_ψ ($\mathcal{T} \rightarrow \mathcal{C}$) using *rule-as-judge* and *MLLM-as-judge*. Some evaluation dimensions are shown as abbreviations for clarity, where *str.*, *sem.*, *sim.*, and *acc.*, represent *visual structure*, *visual semantics*, *visual similarity*, and *visual accuracy*, respectively. Compared to *baselines*, **CoCoEvolve**-enhanced models are optimized at train time and/or test time in *student* mode.

Model	Rule-as-Judge (%)					MLLM-as-Judge (%)					
	str.	sem.	sim.	acc.	M@4	type	data	text	style	visual	M@4
Baselines											
Qwen3-VL-2B	38.06	55.59	54.37	36.47	46.12	36.39	20.56	27.92	18.06	16.67	23.92
Qwen3-VL-4B	51.53	75.22	74.48	54.23	63.87	55.93	33.84	44.81	30.42	27.92	38.58
InternVL3.5-4B	39.10	56.35	54.37	37.68	46.88	35.97	23.94	29.35	15.42	16.25	24.19
Llama3.2-3B	14.71	20.95	20.31	12.29	17.07	12.31	10.15	8.54	6.32	5.47	8.56
DeepSeek-1.3B	30.75	44.62	42.96	26.83	36.29	25.03	18.32	17.97	9.47	10.46	16.25
DeepSeek-6.7B	44.53	63.95	62.29	42.75	53.38	43.06	31.56	32.18	19.04	20.58	29.28
Qwen3-1.7B	36.69	54.28	53.14	35.22	44.83	37.77	23.04	26.19	16.88	17.09	24.19
Qwen3-4B	51.36	73.26	72.27	50.28	61.79	54.70	34.66	35.96	23.61	25.50	34.89
Qwen3-8B	54.60	78.65	77.66	55.72	66.66	61.32	38.38	43.68	27.84	29.11	40.07
CoCoEvolve (@Test)											
Qwen3-VL-2B	58.76	83.36	81.80	58.58	70.63	66.85	37.04	52.96	37.04	32.59	45.30
Qwen3-VL-4B	63.11	90.27	89.64	68.67	77.92	85.56	56.11	71.48	54.44	47.22	62.96
InternVL3.5-4B	62.84	88.01	85.51	61.89	74.56	68.15	47.59	58.70	33.52	32.59	48.11
Llama3.2-3B	58.79	80.66	78.98	56.39	68.71	58.52	45.74	46.48	34.63	28.15	42.70
DeepSeek-1.3B	67.22	91.95	89.94	63.97	78.27	76.48	55.93	57.78	34.27	34.44	51.78
DeepSeek-6.7B	70.26	95.36	93.71	73.60	83.23	88.52	68.15	74.63	49.81	46.48	65.52
Qwen3-1.7B	65.60	90.47	92.31	69.89	79.57	87.96	64.44	70.74	48.89	45.93	63.59
Qwen3-4B	68.94	95.47	94.89	74.48	83.45	95.74	75.74	74.44	55.74	53.52	71.04
Qwen3-8B	71.22	96.80	96.51	77.38	85.48	97.04	77.41	85.37	58.52	55.93	74.85
CoCoEvolve (@Train)											
Qwen3-VL-2B	46.80	60.89	61.39	42.83	52.98	40.03	26.78	29.18	21.32	20.16	27.49
Qwen3-VL-4B	59.05	82.02	81.24	60.32	70.66	59.40	39.81	45.63	33.33	31.39	41.91
Llama3.2-3B	56.04	78.43	74.12	49.57	64.54	38.11	27.86	31.67	13.01	17.19	25.57
DeepSeek-1.3B	63.12	88.03	84.29	58.92	73.59	50.68	32.13	43.00	13.77	22.80	32.48
Qwen3-1.7B	49.85	72.59	70.79	48.05	60.32	49.55	33.54	37.25	21.34	23.28	32.99
Qwen3-4B	60.88	84.82	83.45	60.36	72.38	64.66	38.94	46.86	25.99	30.89	41.47
Qwen3-8B	61.83	86.21	85.34	62.89	74.07	68.93	42.37	48.97	30.96	32.78	44.80
CoCoEvolve (@Train + @Test)											
Qwen3-VL-2B	61.32	89.36	88.38	63.37	75.61	72.37	52.89	60.38	45.83	40.76	54.45
Qwen3-VL-4B	67.52	93.86	93.07	70.39	81.21	88.15	61.67	71.30	55.37	49.44	65.19
Llama3.2-3B	69.86	95.55	91.90	69.07	81.60	74.26	59.81	60.00	32.78	35.19	52.41
DeepSeek-1.3B	69.93	94.84	92.59	71.61	82.24	82.22	61.30	73.71	35.26	41.67	58.83
Qwen3-1.7B	70.18	96.26	95.35	73.41	83.80	89.63	72.96	77.59	55.74	50.37	69.26
Qwen3-4B	71.10	96.31	95.48	75.50	84.60	98.33	76.81	78.15	56.85	54.07	72.84
Qwen3-8B	73.75	98.03	97.21	78.56	86.89	98.98	78.74	88.62	59.26	56.69	76.46

and (2) *out-of-domain tasks*: multi-level complexity tasks on *Chart2Code* (Tang et al., 2025) and *ChartMimic* (Yang et al., 2025b) (§C.3).

Model. We employ three MLLMs and six LLMs as M_θ and M_ψ, including `Qwen3-VL` (2B & 4B) and `InternVL3.5` (4B), `Qwen3` (1.7B, 4B, 8B), `DeepSeek-Coder` (1.3B & 6.7B), and `Llama-3.2` (3B) (Yang et al., 2025a; Wang et al., 2025; Guo et al., 2024; Grattafiori et al., 2024). For evaluation, we leverage `GPT-5-mini` and `Gemini-3-Pro` as LLM and MLLM judges (§3.5 & Fig. 12).

Implementation Details. **CoCoEvolve** trains models for $N_{epoch} = 2$ epochs across $N_{train} = 200$ steps, with an initial learning rate $lr = 1 \times 10^{-6}$ and weighted decay 1×10^{-2} via `AdamW`. We configure $K_\theta = K_\psi = 4$, with global batch size $B = 8$. Our configuration details are in §E.

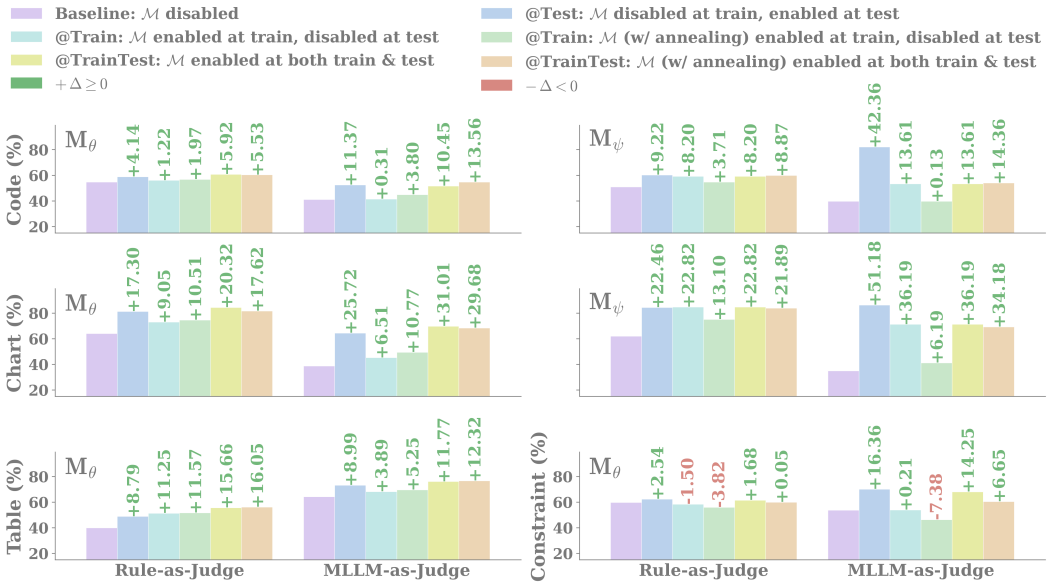


Figure 3: **Ablation Study on Teacher Module.** Capable of seamlessly generalizing to supervised learning, we study the effects of $\mathcal{M}$ (§3.4) in guiding cross-representation understanding.

4.2 Cross-Representation Learning

CoCoEvolve Improves Cross-Representation Understanding Across $\mathcal{V} \Leftrightarrow \mathcal{C} \Leftrightarrow \mathcal{T}$ Representation Cycle. As shown in Tab. 1, CoCoEvolve consistently improves code quality across M_θ and M_ψ, achieving up to 100.00% sandbox execution succuss with **CoCoEvolve@Train+@test**, together with gains of up to $\Delta = \uparrow 17.48\%$ and $\Delta = 26.57\%$ in rule-as-judge and LLM-as-judge evaluations, respectively. Chart evaluation further showcases robust improvements, yielding gains of $\Delta \geq 6.79\%$ and $\Delta \geq 3.33\%$ across rule-as-judge and MLLM-as-judge metrics, respectively. Moreover, table evaluation (Fig. 5) also reveals stable gains through CoCoEvolve, showcasing enhanced performance with up to $\Delta = 15.64\%$ and $\Delta = 24.72\%$ in rule-as-judge and MLLM-as-judge, respectively. Collectively, these evaluations demonstrate that CoCoEvolve effectively strengthens multifaceted cross-representation understanding capabilities of various models across chart, table, and code.

4.3 Generalizability & Robustness

Teacher Module with High-Quality Supervision Facilitates Cross-Representation Learning. Equipped with $\mathcal{M}$, CoCoEvolve is naturally generalizable to supervised learning. Using outcome-verified supervision to guide optimization (§C), we study the effect of $\mathcal{M}$ during train and test time. As shown in Fig. 3, enabling $\mathcal{M}$ only at test time consistently improves performance across all six cross-representation tasks, yielding gains of $\Delta \geq 2.54\%$ across representations. This finding suggests that $\mathcal{M}$ with high-quality supervision can effectively promote transferable cross-representation understanding. Nevertheless, performance becomes less stable when $\mathcal{M}$ is enabled only during training. In contrast, enabling $\mathcal{M}$ at both training and test time yields more balanced overall performance among all cross-representation tasks, indicating that supervision learned during training and teacher-assisted inference complement each other in improving cross-representation reasoning.

CoCoEvolve Improves Generalizability to Out-of-Domain Tasks. Leveraging Chart2Code and Chart2Mimic with varying tasks and complexity levels (§4.1), Fig. 9 demonstrates that CoCoEvolve generalizes effectively to out-of-domain settings, extending beyond direct chart→code reproduction to more challenging scenarios involving multi-input chart reproduction and modification. On ChartMimic, CoCoEvolve improves the performance of both M_θ and M_ψ by up to $\uparrow 37.97\%$ on chart→code reproduction, and up to $\uparrow 34.65\%$ on chart+data→code reproduction. Likewise, on Chart2Code, CoCoEvolve yields substantial gains across all complexity levels and out-of-domain tasks, achieving improvements of up to $\uparrow 37.91\%$ on chart→code reproduction, $\uparrow 46.88\%$ on chart+figure→code modification, $\uparrow 27.85\%$ on chart+table→code modification, and $\uparrow 45.44\%$ on

chart+instruction→code modification. Collectively, these results reveal the robust generalizability of CoCoEvolve to out-of-domain tasks with varying complexity and distribution shifts.

CoCoEvolve Improves Generalizability to Multi-Domain Cross-Representation Understanding. Finetuned on 10,298 instances adapted exclusively from ChartCoder (§C), CoCoEvolve exhibits strong generalizability to multi-domain cross-representation understanding. As shown in Fig. 9, in addition to achieving $\Delta \geq 2.41\%$ on the non-overlapping test set of ChartCoder (Tab. 3), CoCoEvolve further transfers effectively to data across various domains, yielding improvements of up to $\uparrow$ 35.68% on ChartNet and $\uparrow$ 37.97% on ChartMimic. Together, these results demonstrate the robust cross-domain generalization of CoCoEvolve across diverse tasks and representations.

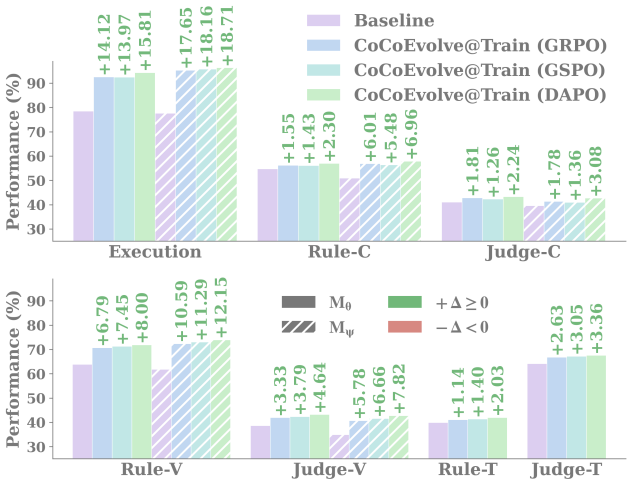


Figure 4: **Generalizability to Different RL Algorithms.** Rule- and Judge- indicate if they are rule-based or LLM- / MLLM- based judges.

CoCoEvolve Integrates Seamlessly with Different RL Algorithms. Through consistency-driven optimization (§3), we integrate CoCoEvolve into GRPO, DAPO, and GSPO and evaluate its effectiveness across M_θ=`Qwen3-VL-4B` and M_ψ=`Qwen3-4B` (§3.2). As shown in Fig. 4, CoCoEvolve consistently improves the cross-understanding performance of both M_θ and M_ψ across all six tasks in the representation cycle, irrespective of the underlying RL algorithm (Δ up to $\uparrow$ 18.71% in execution, $\uparrow$ 6.96% in code, $\uparrow$ 12.15% in chart, $\uparrow$ 3.36% in table). Among evaluated algorithms, DAPO yields the most systematic improvements in cross-representation understanding.

5 Conclusions

In this work, we introduce CoCoEvolve, a **cross-representation consistency-driven co-evolve framework** for structured multimodal reasoning with *self-evolving training dynamic*: as one model improves, it provides stronger supervision to the other, gradually refining both. More broadly, our approach demonstrates that cross-modal consistency can serve as a scalable and reliable supervision signal, offering an alternative to annotation-heavy pipelines for structured reasoning tasks.

AI use statement

In this work, we used generative AI tools to *clean and reformat dataset*. We have **not** used generative AI tools to: *help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, interpret results*. And *generate synthetic data sets, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation, support qualitative and thematic data analysis* are not applicable to this work. Additionally, we used generative AI tools for *correct grammar and polish writing*. We have reviewed all AI-assisted work. For example, we manually go through all the corrected grammar errors to ensure they are not false alarms. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work studies CoCoEvolve in co-evolving two models for cross-representation understanding, and raises no concerns involving human subjects, personal data, or sensitive attributes.

Reproducibility statement

We propose **CoCoEvolve** in full in §3, with an end-to-end elaboration in §D. We also discuss our data preparation details and data statistics in §C, and every metric used to grade a task is defined in §F. The experiment setup and models are discussed in §4.1. We also elaborate on our implementation details in §E, and show our detailed experiment configurations in Tab. 6. Code for the method, the dataset construction, and the evaluation suite will be made public upon acceptance.

References

David B. Adrian, Andras Gabor Kupcsik, Markus Spies, and Heiko Neumann. Cycle-correspondence loss: Learning dense view-invariant visual features from unlabeled and unordered rgb images, 2024. URL https://arxiv.org/abs/2406.12441.

Richard J Chen and Rahul G Krishnan. Self-supervised vision transformers learn visual concepts in histopathology. *Learning Meaningful Representations of Life Workshop, NeurIPS 2021*, 2021.

Inc. Docker. Docker engine, 2013. URL https://www.docker.com/.

Hongchao Fang and Pengtao Xie. An end-to-end contrastive self-supervised learning framework for language understanding. *Transactions of the Association for Computational Linguistics*, 10:1324–1340, 2022. doi: 10.1162/tacl_a_00521. URL https://aclanthology.org/2022.tacl-1.76/.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. Codebert: A pre-trained model for programming and natural languages, 2020. URL https://arxiv.org/abs/2002.08155.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas

Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. Unixcoder: Unified cross-modal pre-training for code representation, 2022. URL https://arxiv.org/abs/2203.03850.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

Xuehang Guo, Zhiyong Lu, Tom Hope, and Qingyun Wang. Anagent for enhancing scientific table & figure analysis, 2026. URL https://arxiv.org/abs/2602.10081.

Kung-Hsiang Huang, Hou Pong Chan, Yi R. Fung, Haoyi Qiu, Mingyang Zhou, Shafiq Joty, Shih-Fu Chang, and Heng Ji. From pixels to insights: A survey on automatic chart understanding in the era of large foundation models, 2024. URL https://arxiv.org/abs/2403.12027.

JaidedAI. Easyocr, 2020. URL https://www.jaided.ai/.

Jovana Kondic, Pengyuan Li, Dhiraj Joshi, Isaac Sanchez, Ben Wiesel, Shafiq Abedin, Amit Alfassy, Eli Schwartz, Daniel Caraballo, Yagmur Gizem Cinar, Florian Scheidegger, Steven I. Ross, Daniel Karl I. Weidele, Hang Hua, Ekaterina Arutyunova, Roei Herzig, Zexue He, Zihan Wang, Xinyue Yu, Yunfei Zhao, Sicong Jiang, Minghao Liu, Qunshu Lin, Peter Staar, Luis Lastras, Aude Oliva, and Rogerio Feris. Chartnet: A million-scale, high-quality multimodal dataset for robust chart understanding, 2026. URL https://arxiv.org/abs/2603.27064.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghua Lin, Anton Ragni, Emmanouil Benetos, Norbert Gyenge, Roger Dannenberg, Ruibo Liu, Wenhu Chen, Gus Xia, Yemin Shi, Wenhao Huang, Yike Guo, and Jie Fu. Mert: Acoustic music understanding model with large-scale self-supervised training, 2024.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Fangyu Liu, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. DePlot: One-shot visual language reasoning by plot-to-table translation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10381–10399, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.660. URL https://aclanthology.org/2023.findings-acl.660/.

Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. ChartAssistant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7775–7803, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.463. URL https://aclanthology.org/2024.findings-acl.463/.

OpenAI. Openai gpt-5, 2025. URL https://openai.com/gpt-5/.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. Codebleu: a method for automatic evaluation of code synthesis, 2020. URL https://arxiv.org/abs/2009.10297.

Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.*, 24:513–523, 1988. URL https://api.semanticscholar.org/CorpusID:7725217.

Zhihong Shao, Peiyi Wang, Yuxiang Wang, and et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Zhanming Shen, Hao Chen, Yulei Tang, Shaolin Zhu, Wentao Ye, Xiaomeng Hu, Haobo Wang, Gang Chen, and Junbo Zhao. CYCLE-INSTRUCT: Fully seed-free instruction tuning via dual self-training and cycle consistency. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5123–5137, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.258. URL https://aclanthology.org/2025.emnlp-main.258/.

Dong Shu, Haoyang Yuan, Yuchen Wang, Yanguang Liu, Huopu Zhang, Haiyan Zhao, and Mengnan Du. Finchart-bench: Benchmarking financial chart comprehension in vision-language models, 2025. URL https://arxiv.org/abs/2507.14823.

Rickard Stureborg, Dimitris Alikaniotis, and Yoshi Suhara. Large language models are inconsistent and biased evaluators, 2024. URL https://arxiv.org/abs/2405.01724.

Jiahao Tang, Henry Hengyuan Zhao, Lijian Wu, Yifei Tao, Dongxing Mao, Yang Wan, Jingru Tan, Min Zeng, Min Li, and Alex Jinpeng Wang. From charts to code: A hierarchical benchmark for multimodal models, 2025. URL https://arxiv.org/abs/2510.17932.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025. URL https://arxiv.org/abs/2508.18265.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. Charxiv: Charting gaps in realistic chart understanding in multimodal llms, 2024b. URL https://arxiv.org/abs/2406.18521.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, Gongye Liu, Xiaomei Nie, Deng Cai, and Yujiu Yang. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation, 2025b. URL https://arxiv.org/abs/2406.09961.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ChartCoder: Advancing multimodal large language model for chart-to-code generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7333–7348, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.363. URL https://aclanthology.org/2025.acl-long.363/.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

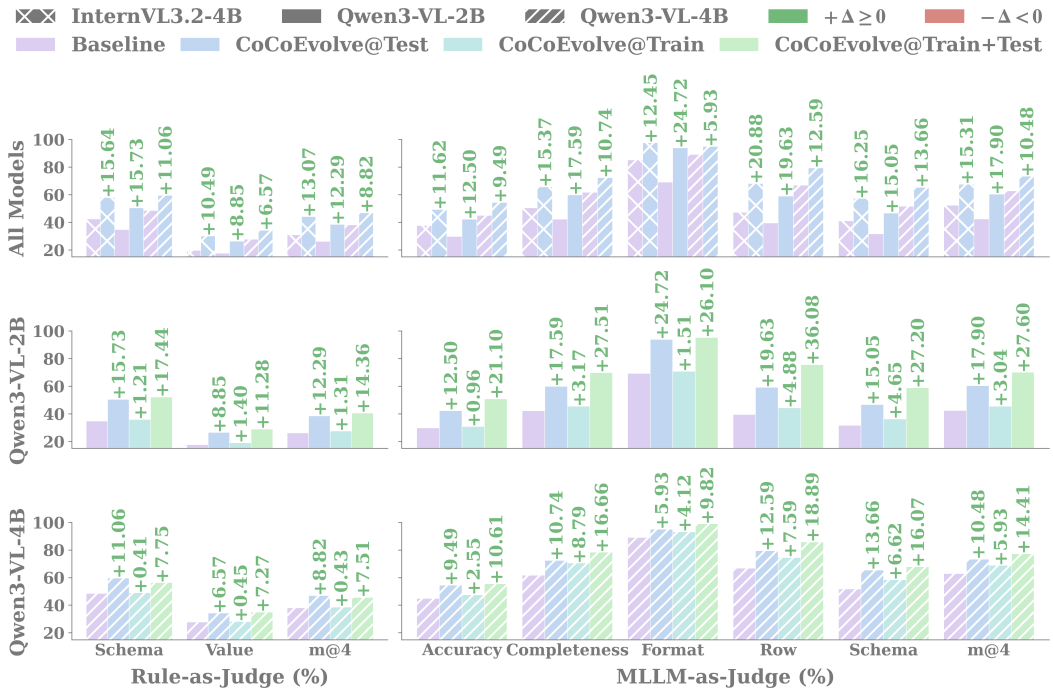


Figure 5: **Performance Evaluation on Table.** We evaluate table performance for M_θ ($\mathcal{V} \rightarrow \mathcal{T}$ and $\mathcal{C} \rightarrow \mathcal{T}$) using *rule-as-judge* and *MLLM-as-judge*. The *data accuracy*, *coverage completeness*, and *row alignment* evaluation dimensions are represented respectively as *acc.*, *coverage*, and *row* for clarity. Compared to *baselines*, **CoCoEvolve**-enhanced models are optimized at train time and/or test time in *student* mode.

A Limitations

In this work, we propose **CoCoEvolve**, a consistency-driven co-evolution framework for optimizing cross-presentation understanding (§3). While **CoCoEvolve** demonstrates promising results in annotation-free cross-representation learning, several limitations remain. First, our current framework instantiates M_θ and M_ψ with small-size model architectures, exploring broader families of MLLMs and LLMs with larger sizes (*e.g.*, 480B) may reveal how architectural choices interact with the co-evolution dynamic and further improve performance. Second, **CoCoEvolve** currently operates on static chart images, while its applicability to dynamic or animated visualizations, where temporal relationships across frames introduce additional representational complexity beyond the *chart-table-code* cycle (§3.1), is underexplored. Extending **CoCoEvolve** to animated cross-representation understanding, where visual, tabular, and code representations must jointly account for temporal semantics, remains an exciting direction for future work.

B Preliminaries

B.1 Can Assumed One-to-One Annotations Reliably Supervise Model Learning?

Q1: Can Assumed One-to-One Annotations Reliably Supervise Model Learning?

Existing cross-representation benchmarks assume *one-to-one* chart-table-code mappings despite their inherently *one-to-many* nature. Can an MLLM trained on such ill-defined *unconstrained one-to-one* annotations achieve optimized performance?

To answer Q1, we finetune `Qwen3-VL-4B` (M_θ) and `Qwen3-4B` (M_ψ) via SFT, finetuned for 1 epoch on 30,000 instances randomly sampled from *ChartNet* (Kondic et al., 2026), a benchmark that

assumes *one-to-one* chart-table-code correspondences. We then evaluate both the base models and the SFT-finetuned M_θ and M_ψ on the same held-out test set of 542 non-overlapping instances randomly sampled from the same benchmark (Tab. 3).

As shown in Fig. 6, we evaluate M_θ and M_ψ under three settings: non-finetuned baseline, SFT, and non-finetuned **CoCoEvolve@Test**, using the same non-overlapping test set adapted from ChartNet. Although the SFT-trained M_θ, finetuned on additional training instances, achieves slightly improved performance on chart-to-code generation ($\Delta =\uparrow 1.14\%$ in rule-as-judge code evaluation, $\Delta =\uparrow 4.57\%$ in LLM-as-judge code evaluation), it exhibits noticeably degraded performance on code, chart, and table evaluation across sandbox execution ($\Delta =\downarrow 11.31\%$), rule-as-judge (up to $\Delta =\downarrow 7.85\%$), and MLLM-as-judge (up tp $\Delta =\downarrow 8.48\%$). Moreover, not only does M_θ perform worse on chart and table evaluation, but M_ψ also shows significant degradation across chart, code, and table evaluation (up to $\Delta =\downarrow 21.25\%$). In addition, both M_θ and M_ψ exhibit significantly reduced code execution success rates (up to $\Delta =\downarrow 14.93\%$). These results reveal that the assumed one-to-one annotations, *without* properly defined constraints, are inherently one-to-many and thus ill-defined and unreliable for guiding and supervising model learning (§1). This motivates the critical need for a principled constraint definition to properly ground *one-to-many* mappings to precise *one-to-one* correspondences (§3.1).

B.2 Can Fixed Supervision Reliably Guides Cross-Representation Learning?

Q2: Can Fixed Supervision Reliably Guides Cross-Representation Learning?

Even with principled constraints (§3.1), can an MLLM trained on fixed *chart-to-table* and *chart-to-code* pairs not only achieve robust performance on unseen instances of trained tasks (*chart-to-table* and *chart-to-code*), but also reliably generalize to held-out tasks, directions, and representation combinations?

To answer *Q2*, we further finetune `Qwen3-VL-4B` and `Qwen3-4B` via SFT (1 epoch) on exactly the same number of *fixed* yet *constrained* (§3.1) *chart-to-table* and *chart-to-code* pairs as in our main experiments (§C & Tab. 3). Note that while our main experiments use only *chart-to-code* instances adapted from the randomly sampled training subset of the original benchmark, this preliminary study additionally includes fixed *chart-to-table* annotations. Since ground-truth tables are unavailable, we use `Gemini-3-Pro` to generate the reference table and constraint through outcome-grounded filtering (§C). These MLLM-generated tables are used for both training and evaluation in this study (Tab. 3).

Next, we evaluate M_θ and M_ψ under three settings: non-finetuned baseline, SFT, and non-finetuned **CoCoEvolve@Test**, using the same non-overlapping test set across *six* cross-representation tasks (§3.1 & §4.1), covering trained tasks ($\mathcal{V} \Leftrightarrow \mathcal{T}$, $\mathcal{V} \Leftrightarrow \mathcal{C}$), reversed directions ($\mathcal{T} \Leftrightarrow \mathcal{V}$, $\mathcal{C} \Leftrightarrow \mathcal{V}$), and unseen representation combinations ($\mathcal{T} \Leftrightarrow \mathcal{C}$). Results in Fig. 7 demonstrate that fixed supervision signals not only fail to reliably optimize M_θ and M_ψ on their trained tasks (with Δ up to $\downarrow 25.19\%$), but are also unable to generalize to reversed directions (with Δ up to $\downarrow 25.10\%$) and unseen representation combinations (with Δ up to $\downarrow 31.82\%$), motivating the need for a principled, task-agnostic, and direction-agnostic supervision paradigm as proposed in **CoCoEvolve** (§3).

B.3 How Do Models Fail in Cross-Representation Understanding?

Motivated by our findings on the substantial unreliability of unconstrained cross-representation understanding (§B.1) and the limitations of fixed supervision signals (§B.2), we conduct a third preliminary study to qualitatively analyze *how* models fail during cross-representation understanding.

Fig. 8 shows a representative failure example from our case studies. Given a chart image, we analyze the model outputs across multiple representations, including structured tables, visualization code, and rendered charts. We observe two key failure patterns:

1. **Model failures are highly correlated across representations.** When the model fails to understand a chart, similar failures consistently appear across its generated representations. For example, incorrect numerical values, missing titles and labels, and incomplete information extraction observed in the table are also reflected in the generated code and rendered visualization. Different representations tend to exhibit similar understanding errors despite expressing infor-

mation in different forms, suggesting that model failures originate from a shared deficiency in underlying chart understanding rather than isolated representation-specific errors. When situated in the representation cycle (§3.1), such failures propagate rapidly across different representations, resulting in accumulated errors and increasingly amplified inaccuracies.

2. **Representations remain highly inconsistent despite describing the same semantics.** Although different representations are expected to encode the same chart semantics, the generated outputs frequently disagree. In Fig. 8, the extracted table only contains information from the left subplot, while the generated visualization code attempts to reconstruct both subplots with substantial inaccuracies. Similar inconsistencies appear in rendered outputs through mismatched metadata, incorrect scales, distorted layouts, and inconsistent visual encodings. These observations suggest that existing models struggle to maintain semantic agreement across representations.

Collectively, our analysis reveals that cross-representation failures are characterized not only by *low fidelity within individual representations*, but also by *poor agreement among representations*. This indicates that optimizing representations independently or relying on fixed supervision signals are insufficient for robust cross-representation understanding.

These findings motivate our **consistency-driven co-evolution framework** (§3). Instead of leveraging unconstrained or fixed supervision signals (§B.1-B.2), **CoCoEvolve** explicitly encourage agreement across representations and reward mutually consistent generations. By enforcing cross-representation consistency, models co-evolve symbiotically, improving understanding fidelity within each representation and enhancing alignment across representations, thereby optimizing overall cross-representation understanding.

Table 3: **Data Statistics.** We summarize the data statistics of our datasets used in preliminary studies, training, and evaluation. All *adapted* data are randomly sampled from the original benchmarks and adapted to our cross-representation tasks, *sandbox* denotes sandbox execution, and *generated* represents MLLM-generated samples where we use `Gemini-3.1-Pro` to generate the ground-truth table and constraint for each chart instance in the training subset randomly sampled from the original benchmark. ✓(n) denotes the number of fine-grained chart-to-code generation tasks: (a) *ChartCoder*: (1) chart→code (reproduction); (b) *ChartMimic*: (1) chart→code (reproduction), (2) chart+data→code (reproduction); (c) *Chart2Code*: (1) chart→code (reproduction), (2) chart+instruction→code (modification), (3) chart+table→code (modification), (4) chart+figure→code (modification); (d) *ChartNet*: (1) chart→code (reproduction).

Source	$\mathcal{V} \Leftrightarrow \mathcal{C}$		$\mathcal{V} \Leftrightarrow \mathcal{T}$		$\mathcal{T} \Leftrightarrow \mathcal{C}$	
	$\mathcal{V} \to \mathcal{C}$	$\mathcal{C} \to \mathcal{V}$	$\mathcal{V} \to \mathcal{T}$	$\mathcal{T} \to \mathcal{V}$	$\mathcal{T} \to \mathcal{C}$	$\mathcal{C} \to \mathcal{T}$
Preliminary Study - Q1 (*train*: 30,000 | *test*: 542)						
ChartNet (Kondic et al., 2026)	✓ (1)	✓	✓	✓	✓	✓
Data	*adapted*	*sandbox*	*adapted*	*adapted*	*adapted*	*adapted*
N_{train}	30,000	-	-	-	30,000	-
N_{test}	542	-	542	542	542	542
Preliminary Study - Q2 (*train*: 10,298 | *test*: 542)						
ChartCoder (Zhao et al., 2025)	✓ (1)	✓	✓	✓	✓	✓
Data	*adapted*	*sandbox*	*generated*	*generated*	*generated*	*generated*
N_{train}	10,298	-	-	-	-	-
N_{test}	3,411	-	1082	1082	1082	1082
Train (*train*: 10,298)						
ChartCoder (Zhao et al., 2025)	✓ (1)	✓	✓	✓	✓	✓
Data	*adapted*	*sandbox*	-	-	-	-
N_{train}	10,298	-	-	-	-	-
Test (*test*: 3,411)						
ChartCoder (Zhao et al., 2025)	✓ (1)	✓	✓	✓	✓	✓
Data	*adapted*	*sandbox*	*generated*	*generated*	*generated*	*generated*
N_{test}	542	-	542	542	542	542
ChartMimic (Yang et al., 2025b)	✓ (2)	✓	✗	✗	✗	✗
Data	*adapted*	*sandbox*	-	-	-	-
N_{test}	1084	-	-	-	-	-
Chart2Code (Tang et al., 2025)	✓ (4)	✓	✗	✗	✗	✗
Data	*adapted*	*sandbox*	-	-	-	-
N_{test}	1243	-	-	-	-	-
ChartNet (Kondic et al., 2026)	✓ (1)	✓	✓	✓	✓	✓
Data	*adapted*	*sandbox*	*adapted*	*adapted*	*adapted*	*adapted*
N_{test}	542	-	542	542	542	542

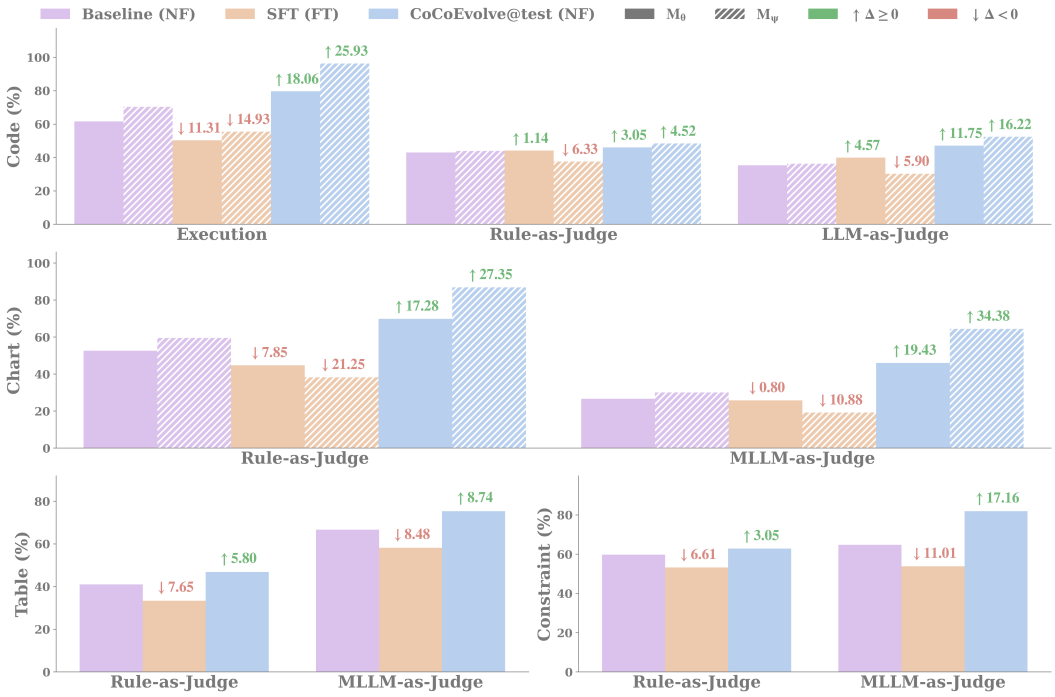


Figure 6: **Significance of Constraints in Cross-Representation Understanding.** Comparing M_θ and M_ψ variants among baseline, non-finetuned (NF) **CoCoEvolve@Test** (§3.3), and SFT (§B.1), SFT-trained M_θ and M_ψ exhibit notably degraded performance, highlighting the significance of constrained one-to-one mapping in cross-representation understanding.

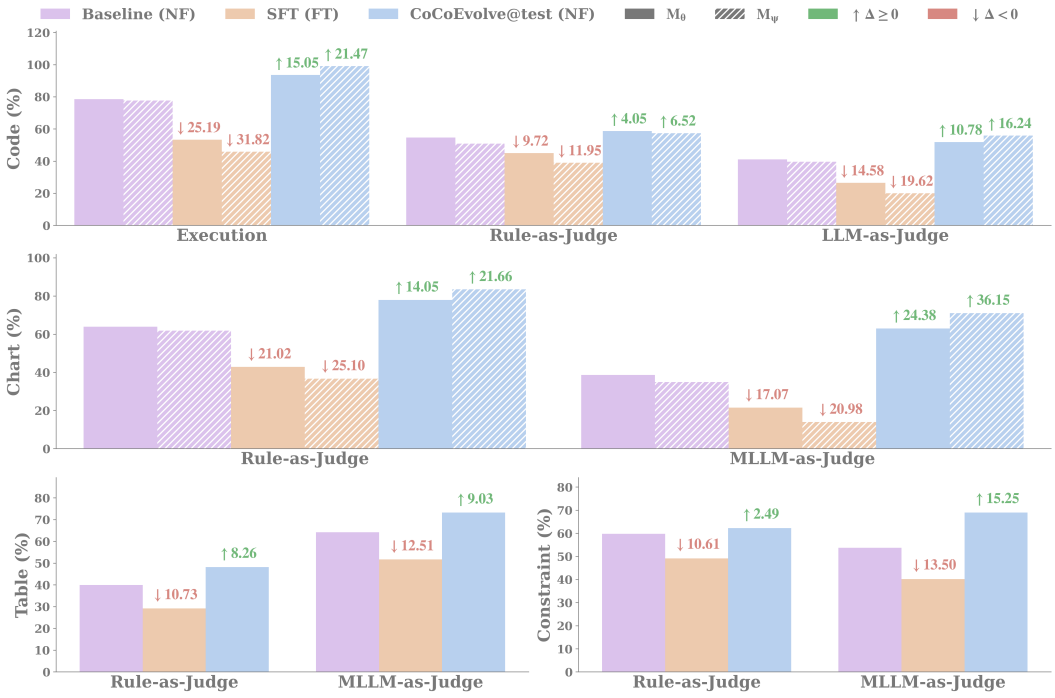
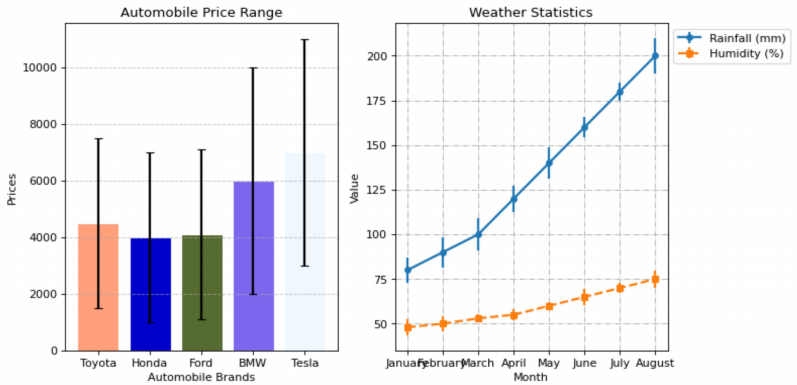


Figure 7: **Limitations of Fixed Supervision Signals in Cross-Representation Understanding.** Comparing M_θ and M_ψ variants among baseline, non-finetuned (NF) **CoCoEvolve@Test** (§3.3), and SFT (§B.1), SFT-trained (FT) M_θ and M_ψ show noticeably degraded performance, revealing the critical limitations of fixed annotations in supervising cross-representation learning.

How Do Models Fail in Cross-Representation Understanding?

Original Chart



$\mathcal{V} \rightarrow \mathcal{T}$

Automobile Brands	Min	Q1	Median	Q3	Max	Outlier
Toyota	1000	2000	4000	6000	7500	[]
Honda	500	1500	3500	5500	7000	[12000]
Ford	750	1800	4000	5000	6500	[]
BMW	1200	2500	4500	6000	8000	[]
Tesla	1500	3000	5000	7000	10500	[]



1. Significant numerical extraction errors: The model exhibits substantial inaccuracies in extracting numerical values from the visualization. For instance, the maximum value for BMW is incorrectly extracted as 8,000 instead of 10,000, and the minimum value for Toyota is underestimated as 1,000 despite clearly closer to 1,500.
2. Loss of unit and attribute alignment: Extracted values are not grounded to their associated semantic units or attributes, causing confusion across subplots (e.g., values are Price in the left subplot versus Rainfall (mm) and Humidity (%) in the right subplot).
3. Cross-representation omission: The model fails to incorporate the entire right subplot into the structured output, resulting in missing textual and numerical information and incomplete cross-representation understanding.

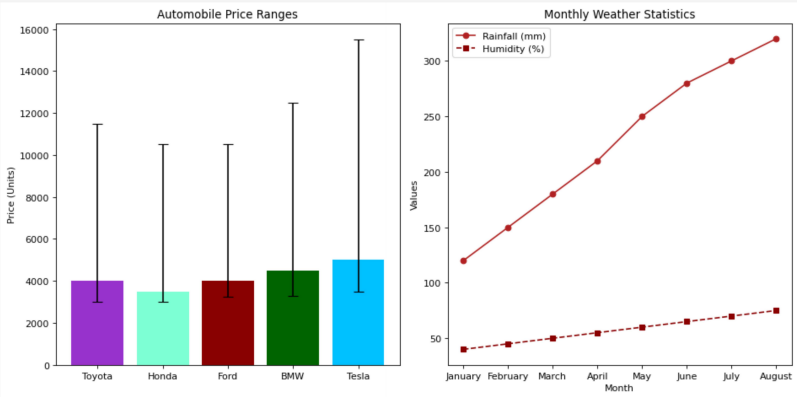
$\mathcal{V} \rightarrow \mathcal{C}$

```
import matplotlib.pyplot as plt
import numpy as np

# Data for the bar chart
brands = ["Toyota", "Honda", "Ford", "BMW", "Tesla"]
min_prices = [1000, 500, 750, 1200, 1500]
q1_prices = [2000, 1500, 1800, 2500, 3000]
median_prices = [4000, 3500, 4000, 4500, 5000]
q3_prices = [6000, 5500, 5000, 6000, 7000]
max_prices = [7500, 7000, 6500, 8000, 10500]

# Data for the line chart
months = ["January", "February", "March", "April", "May", "June", "July", "August"]
rainfall = [120, 150, 180, 210, 250, 280, 300, 320]
humidity = [40, 45, 50, 55, 60, 65, 70, 75]

# Set up the figure and axes
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Bar chart for automobile brands
axs[0].bar(brands, median_prices, color=['#9932CC', '#7FFFD4', '#8B0000', '#006400', '#00BFFF'],
           yerr=[min_prices, max_prices], capsize=5)
axs[0].set_title('Automobile Price Ranges')
axs[0].set_ylabel('Price (Units)')
axs[0].set_xticks(range(len(brands)))
axs[0].grid(False)

# Line chart for rainfall and humidity over months
axs[1].plot(months, rainfall, label='Rainfall (mm)', color='#B22222', linestyle='solid', marker='o')
axs[1].plot(months, humidity, label='Humidity (%)', color='#8B0000', linestyle='dashed', marker='s')
axs[1].set_title('Monthly Weather Statistics')
axs[1].set_xlabel('Month')
axs[1].set_ylabel('Values')
axs[1].legend()

# Adjust layout and save the figure
plt.tight_layout()
plt.savefig('workspace/generated_chart.png', bbox_inches='tight', dpi=80)
```



1. Propagation of data extraction errors: The model-generated visualization code inherits the same failure patterns observed in the intermediate value extraction table, resulting in incorrect numerical values and incomplete data reconstruction.
2. Missing and incorrect metadata encoding: Chart metadata are not faithfully preserved. Specifically, the Y-axis label is incorrectly generated as "Price (Units)" instead of "Price", and the X-axis title is omitted entirely.
3. Inaccurate axis scaling and tick configuration: The Y-axis tick values and range do not match the original visualization, leading to distorted quantitative interpretation.
4. Color encoding inconsistency: The generated chart fails to preserve the original color mapping, producing colors that differ substantially from those in the source visualization.
5. Layout and geometric distortion: The model introduces structural rendering errors by failing to preserve the original subplot geometry. The source figure contains vertically oriented rectangular subplots (width < height), whereas the generated code assumes a figure size of (12, 6), resulting in nearly square subplot layouts and altered visual proportions.

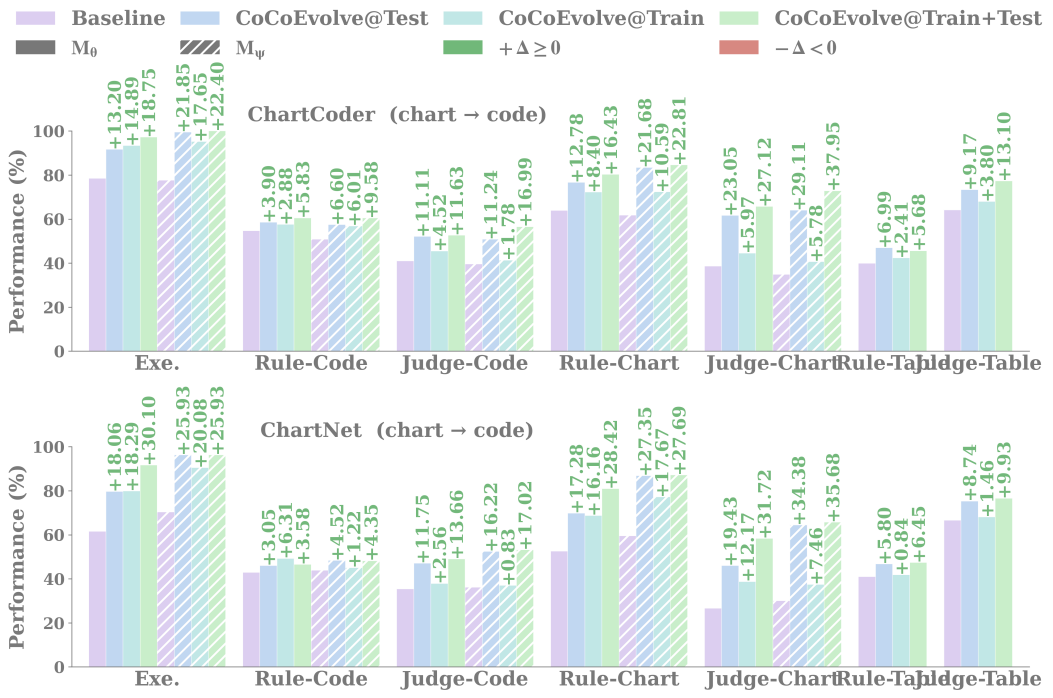
$\mathcal{C} \rightarrow \mathcal{V}$

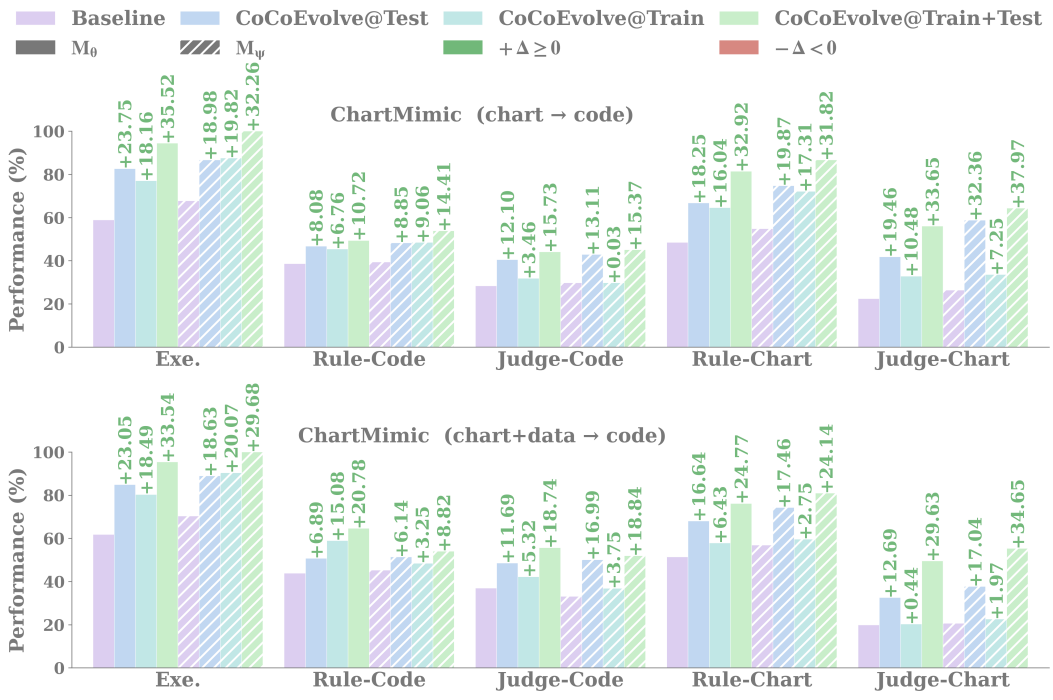




1. Propagation of data extraction errors: The rendered visualization inherits the same failure patterns observed in the intermediate value extraction, leading to incorrect numerical values and incomplete reconstruction of the underlying data.
2. Propagation of code generation errors: The rendered chart further inherits errors introduced during visualization code generation, including missing or incorrect metadata encoding, inaccurate axis scaling and tick configuration, inconsistent color encoding, as well as layout and geometric distortions.
3. Cross-representation inconsistency: The model demonstrates weak alignment across representations. While only the left subplot is retained in the intermediate tabular representation, the generated code reconstructs both subplots, though introducing substantial errors. This inconsistency persists throughout the chart-to-code pipeline, indicating unstable transfer of visual information across representations.

Figure 8: **Cross-Representation Understanding Failures.** A case study from our preliminary analysis illustrating how model errors accumulate and propagate across representations, resulting in inaccurate data extraction, incomplete information transfer, and significant cross-representation inconsistencies.


Baseline
CoCoEvolve@Test
CoCoEvolve@Train
CoCoEvolve@Train+Test
M_θ
M_ψ
$+\Delta \geq 0$
$-\Delta < 0$
ChartCoder (chart → code)
ChartNet (chart → code)
Performance (%)
Exe.
Rule-Code
Judge-Code
Rule-Chart
Judge-Chart
Rule-Table
Judge-Table


Baseline
CoCoEvolve@Test
CoCoEvolve@Train
CoCoEvolve@Train+Test
M_θ
M_ψ
$+\Delta \geq 0$
$-\Delta < 0$
ChartMimic (chart → code)
ChartMimic (chart+data → code)
Performance (%)
Exe.
Rule-Code
Judge-Code
Rule-Chart
Judge-Chart

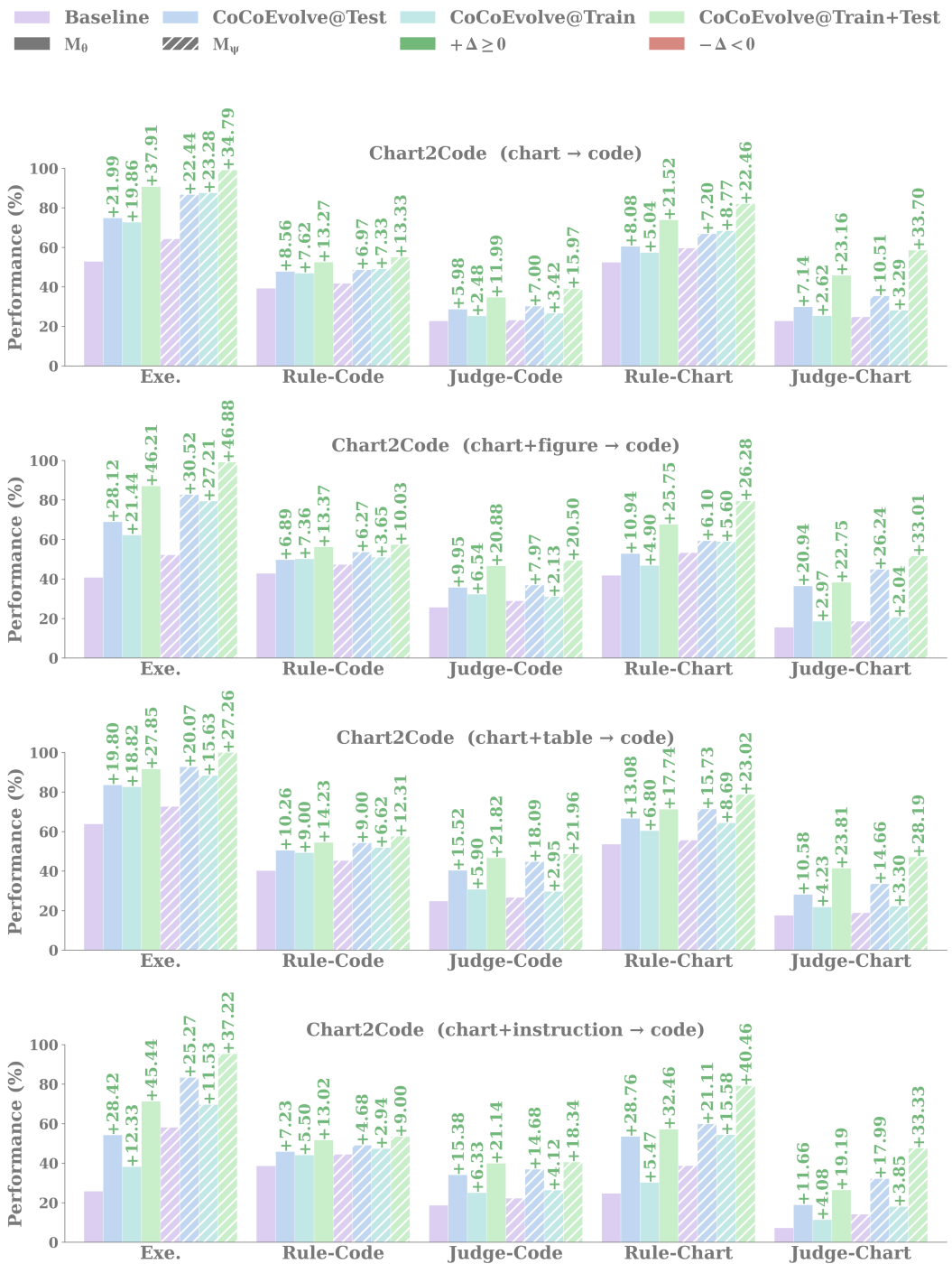


Figure 9: **Out-of-Domain Generalization.** We extend our evaluation to out-of-domain datasets and tasks (§4.1), leveraging `Qwen3-VL-4B` and `Qwen3-4B` as M_θ and M_ψ, respectively.

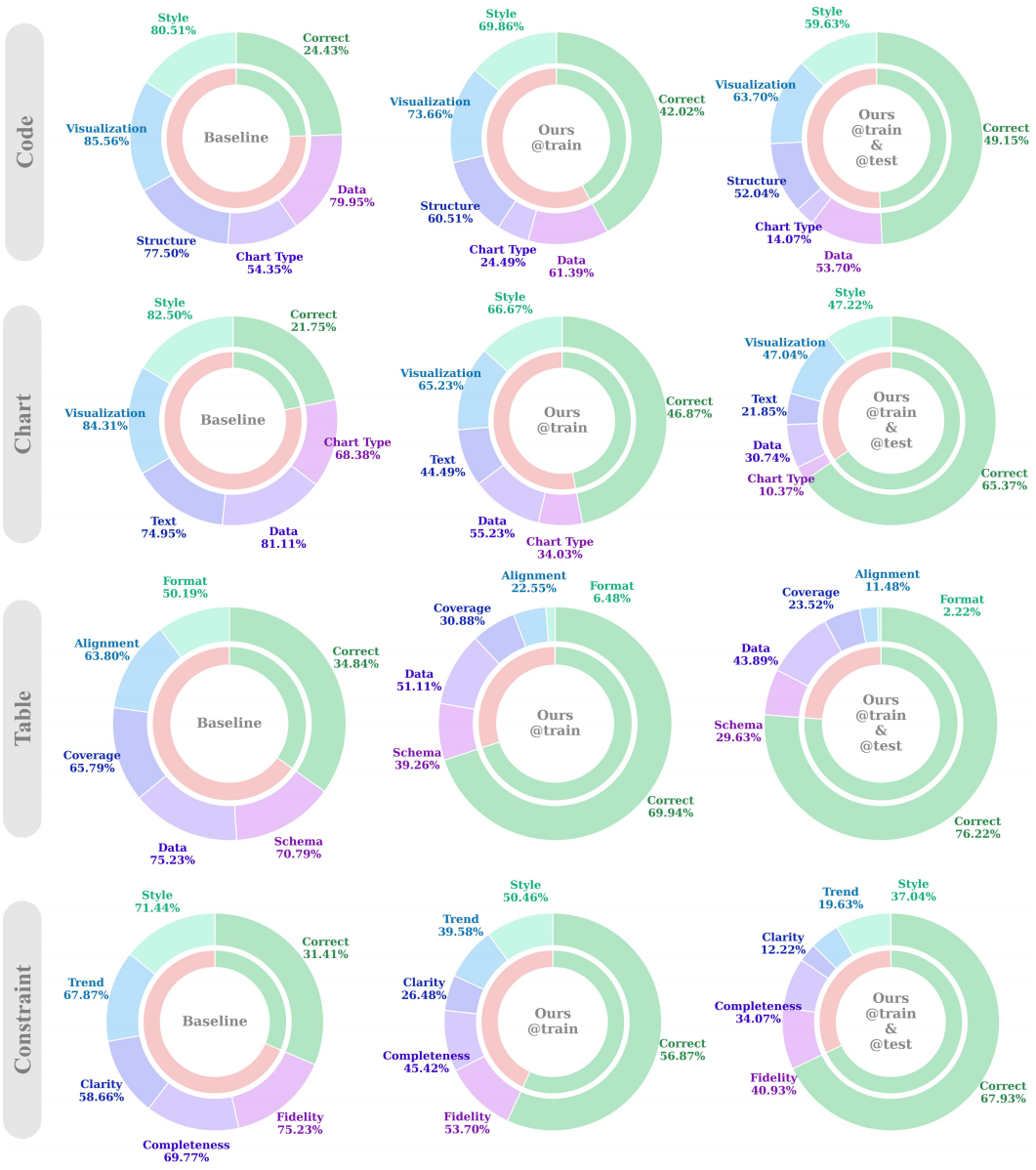


Figure 10: **Failure Analysis.** We visualize the failure distribution of `Qwen3-4B-VL` in cross-representation understanding, where baseline is the non-finetuned base model, ours (@train) is `Qwen3-4B-VL` finetuned through **CoCoEvolve@Train**, and ours (@train & @test) is ours (@train) further optimized by **CoCoEvolve@Test**.

C Dataset Construction

We construct our training and evaluation datasets by adapting existing chart benchmarks to our cross-representation learning framework (§3). All adapted instances are reformatted to our representation cycle instructions (Figs. 14-16) and verified through sandbox execution. Tab. 3 summarizes the full data statistics across our training and evaluation datasets.

C.1 Sandbox Environment

To verify the executability of visualization code across dataset construction, training, and evaluation, we create our sandbox execution environment via Docker (Docker, 2013) pre-installed with a set of commonly used data visualization and scientific computing packages, including `matplotlib`, `seaborn`, `plotly`, `pandas`, `numpy`, `scipy`, `scikit-learn`, `Pillow`, `squarify`, etc. Each code execution is run in an isolated container with a fixed timeout of $l_{timeout}$ seconds, ensuring reproducibility and safety across all dataset construction, training, and evaluation stages. As such, our sandbox reliably powers our deterministic code executor h (Eq. 1), where $h(\hat{c})$ returns the rendered chart image on success and fails otherwise.

C.2 Training Dataset

We construct our training dataset exclusively from ChartCoder (Zhao et al., 2025) through three filtering stages followed by a final sampling step:

Stage 1 on Filtering: Sandbox Executability. We filter instances whose reference code renders successfully via our deterministic executor h (§3.1), ensuring all training instances are verifiably executable in our sandbox environment.

Stage 2 on Filtering: Token Length. We filter instances satisfying $\texttt{len}(\textit{input}) \leq l_{\textit{input}}$ and $\texttt{len}(c) \leq l_{\textit{code}}$, where $l_{\textit{input}}$ and $l_{\textit{code}}$ are maximum token length thresholds for the chart input and reference code, respectively.

Stage 3 on Filtering: Outcome-Grounded Verification. For each remaining instance, we generate candidate reference tables t_{gt} and constraints s_{gt} using `GPT-5-mini` (OpenAI, 2025), and retain only instances where the generated t_{gt} and s_{gt} verifiably improve downstream M_{ψ} (powered by `Qwen3-4B`) code generation performance, ensuring their quality as reliable evaluation references. We further filter instances satisfying $\texttt{len}(t_{gt}) \leq l_{\textit{table}}$ and $\texttt{len}(s_{gt}) \leq l_{\textit{constraint}}$, where $l_{\textit{table}}$ and $l_{\textit{constraint}}$ are maximum token length thresholds for the reference table and constraint, respectively.

Stage 4 on Sampling. From the filtered dataset, we randomly downsample 10,298 instances as our final training set. From the remaining instances, we further randomly sample 542 non-overlapping instances as the in-domain evaluation set for the ChartCoder benchmark (Zhao et al., 2025) adapted to our cross-representation understanding tasks.

C.3 Evaluation Datasets

We construct evaluation datasets from four benchmarks through the same filtering pipeline as the training dataset, adapting each to cross-representation learning instructions. All evaluation instances are non-overlapping with the training set.

Stages 1-3 on Filtering. All evaluation instances undergo the same three filtering stages as the training dataset (§C.2): (1) sandbox executability verification via h; (2) token length filtering with thresholds $l_{\textit{input}}$, $l_{\textit{code}}$; and (3) reference table t_{gt} and constraint s_{gt} generation and outcome-grounded verification, followed by token length filtering with thresholds $l_{\textit{table}}$, $l_{\textit{constraint}}$.

Stage 4 on Sampling. From the filtered instances, we construct the following evaluation sets, all randomly sampled and adapted to our cross-representation learning instructions:

1. **ChartCoder** (Zhao et al., 2025) (542 instances): in-domain evaluation set adapted to our cross-representation understanding tasks, non-overlapping with the training set.

2. **ChartNet** (Kondic et al., 2026) (542 instances): out-of-domain evaluation set adapted to our cross-representation understanding tasks, covering different domains (e.g., health, finance, etc.) and visualization packages (e.g., `matplotlib`, `plotly`, etc.)
3. **ChartMimic** (Yang et al., 2025b) (1084 instances): evaluation set adapted to our cross-representation understanding tasks, comprising 542 in-domain cross-representation understanding tasks and 542 out-of-domain chart modification instances covering diverse scientific domains, such as physics, mathematics, economics, biology, etc.
4. **Chart2Code** (Tang et al., 2025) (1,243 instances): evaluation set adapted to four tasks covering multi-level chart complexity, including cross-representation understanding (*in-domain*), text-based chart modification (*out-of-domain*), table-based chart modification (*out-of-domain*), and figure-based chart modification (*out-of-domain*).

All token length thresholds l_{input}, l_{code}, l_{table}, and $l_{constraint}$, together with sampling configurations, are summarized in §E.1 and Tab. 6.

D CoCoEvolve@Train: Consistency-Driven Co-Evolution

In the co-evolution objective (Eq. 8), each surrogate is a token-level clipped-ratio loss with KL penalty:

$$\mathcal{L}_\theta^{(i)} = \frac{1}{|o_\theta^{(i)}|}\sum_\tau \min\Big(\rho_{\theta,\tau}^{(i)}A_\theta^{(i)},\ \mathrm{clip}(\rho_{\theta,\tau}^{(i)},1-\epsilon,1+\epsilon)A_\theta^{(i)}\Big) - \beta\,\mathbb{D}_{\mathrm{KL}}\big[\pi_\theta \,\|\, \pi_\theta^{\mathrm{ref}}\big] \tag{15}$$

$$\mathcal{L}_\psi^{(i,k)} = \frac{1}{|o_\psi^{(i,k)}|}\sum_\tau \min\Big(\rho_{\psi,\tau}^{(i,k)}A_\psi^{(i,k)},\ \mathrm{clip}(\rho_{\psi,\tau}^{(i,k)},1-\epsilon,1+\epsilon)A_\psi^{(i,k)}\Big) - \beta\,\mathbb{D}_{\mathrm{KL}}\big[\pi_\psi \,\|\, \pi_\psi^{\mathrm{ref}}\big] \tag{16}$$

The coupling is carried by the advantages. Let $r_\theta(o_\theta^{(i)}, o_\psi^{(i,k)}; v)$ and $r_\psi(o_\theta^{(i)}, o_\psi^{(i,k)}; v)$ denote the paired reward functionals defined in Eq.7, where $R_\theta^{(i,k)} = r_\theta(o_\theta^{(i)}, o_\psi^{(i,k)}; v)$ and $R_\psi^{(i,k)} = r_\psi(o_\theta^{(i)}, o_\psi^{(i,k)}; v)$ denote the corresponding scalar rewards. The π_θ advantage marginalizes over π_ψ noise by averaging across its K_ψ children of each π_θ rollout:

$$\bar{r}_\theta^{(i)} = \frac{1}{K_\psi}\sum_{k=1}^{K_\psi} r_\theta\big(o_\theta^{(i)}, o_\psi^{(i,k)}; v\big) \tag{17}$$

$$A_\theta^{(i)} = \frac{\bar{r}_\theta^{(i)} - \mu_\theta(v)}{\sigma_\theta(v)+\delta} \tag{18}$$

so that credit assignment to π_θ reflects expected cross-model quality rather than a single π_ψ draw. The π_ψ advantage is computed within the K_ψ-sibling group conditioned on the same π_θ parent:

$$A_\psi^{(i,k)} = \frac{r_\psi\big(o_\theta^{(i)}, o_\psi^{(i,k)}; v\big) - \mu_\psi^{(i)}}{\sigma_\psi^{(i)}+\delta} \tag{19}$$

where $(\mu_\theta(v), \sigma_\theta(v))$ are the mean and standard deviation of $\{\bar{r}_\theta^{(i)}\}_{i=1}^{K_\theta}$, $(\mu_\psi^{(i)}, \sigma_\psi^{(i)})$ those of $\{r_\psi(o_\theta^{(i)}, o_\psi^{(i,k)}; v)\}_{k=1}^{K_\psi}$, $\rho_{\theta,\tau}^{(i)} = \pi_\theta(o_{\theta,\tau}^{(i)} \mid v, o_{\theta,<\tau}^{(i)})/\pi_{\theta,\mathrm{old}}(\cdot)$ and $\rho_{\psi,\tau}^{(i,k)} = \pi_\psi(o_{\psi,\tau}^{(i,k)} \mid s^{(i)}, t^{(i)}, o_{\psi,<\tau}^{(i,k)})/\pi_{\psi,\mathrm{old}}(\cdot)$ are per-token importance ratios, and δ is a numerical stabilizer.

E Implementation Details

In Tabs. 4-6, we summarize the core symbols, notations, and hyperparameter settings used in this paper. Below, we further explain the key design choices to facilitate reproducibility and adaptation to customized use cases.

E.1 Experiment Configuration

Three-Tier Notation. Our method involves two models interacting through a shared representation cycle, which requires careful notational separation between their functional roles, model identities, and training policies (Tab. 4):

1. f_θ and g_ψ denote *task mappings*: the core representational objectives each model is designed to perform. Note that f_θ formally describes the primary cross-representation objective without enumerating all outputs of M_θ, whose full rollout $o_\theta^{(i)} = (\hat{s}^{(i)}, \hat{t}^{(i)}, \hat{c}_\theta^{(i)})$ additionally includes a constraint and a reference code for co-evolution.
2. M_θ and M_ψ denote *model identities* instantiating these mappings: M_θ is a multimodal model and M_ψ is a language model. They can be instantiated by either the same or different base models, depending on experimental settings (§4).
3. π_θ and π_ψ denote the *training policies* parameterizing M_θ and M_ψ during RL: used specifically in the context of rollout sampling, importance ratio computation, advantage estimation, and policy updates.

This three-tier design allows us to cleanly distinguish what each model *does* (f_θ, g_ψ), *what* it *is* (M_θ, M_ψ), and *how* it is *optimized* (π_θ, π_ψ).

Asymmetric Reward Weighting. All reward weighting coefficients $\lambda_*^{(\pi)}$ and sub-weights $\omega_*^{(\pi)}$ are defined per-model, and their values in our main experiments are intentionally asymmetric between M_θ and M_ψ (Tab. 6). This asymmetry reflects the fundamentally different roles the two models play in the representation cycle. Generating code, table, and constraint from a given chart, the primary challenges of M_θ lie in perceptual grounding and semantic understanding. We therefore assign M_θ a higher individual grounding weight $\lambda_\theta = 0.7$ while keeping its cross-model consistency weights lower ($\lambda_c^{(\theta)} = 0.05$, $\lambda_v^{(\theta)} = 0.05$), ensuring that M_θ's training signal is dominated by how well its understanding matches the original chart v, rather than how well it agrees with M_ψ. By contrast, M_ψ performs single-modal understanding, with its primary challenge being the faithful translation of constrained table data into the chart it originally corresponds to. As such, cross-model consistency is the primary grounding signal ofM_ψ. We therefore assign M_ψ higher cross-model consistency weights ($\lambda_c^{(\psi)} = 0.2$, $\lambda_v^{(\psi)} = 0.2$) to reflect this. The same asymmetry applies to the code consistency sub-weights $\omega_e^{(\pi)}$ and $\omega_s^{(\pi)}$: M_θ is assigned higher execution weight ($\omega_e^{(\theta)} = 0.9$) since binary execution success is a stronger and more direct signal for its cross-representation learning objectives, while M_ψ receives a more balanced split ($\omega_e^{(\psi)} = 0.7$, $\omega_s^{(\psi)} = 0.3$) that additionally rewards semantic code similarity. Importantly, these asymmetric weights also serve as an anti-collusion mechanism to avoid reward hacking and training crash: by assigning M_θ stronger individual grounding rewards relative to cross-model consistency rewards, we prevent the degenerate solution where both models collude to produce mutually consistent but semantically incorrect outputs (Fig. 11).

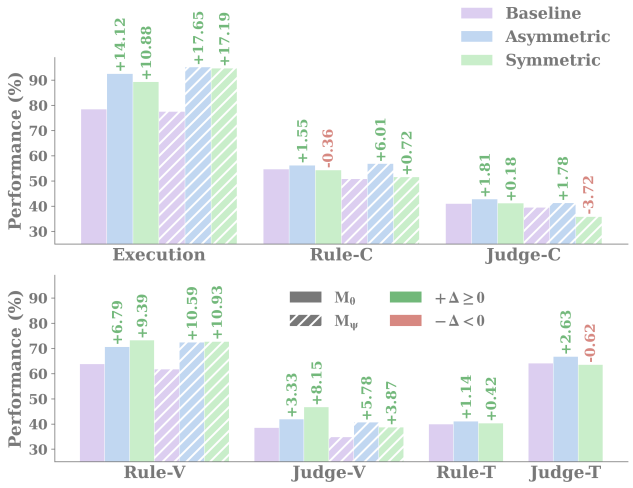


Figure 11: **Ablation Study on Reward Weighting.**

Weighting Coefficient Families. We use three distinct families of weighting coefficients for different stages of **CoCoEvolve**, each serving a specialized purpose:

1. $\omega_*^{(\pi)}$ denotes sub-weights *within* a reward component, balancing the contribution of each constituent metric.
2. $\lambda_*^{(\pi)}$ denotes per-model weights combining heterogeneous reward components into the final reward R_π during training, with asymmetric configurations across M_θ and M_ψ reflecting their distinct roles in tackling the representation cycle (§3.2).
3. γ_* denotes weights combining individual evaluation metrics into final composite scores during evaluation, enabling comprehensive multidimensional assessment across all six cross-representation tasks (§3.5).

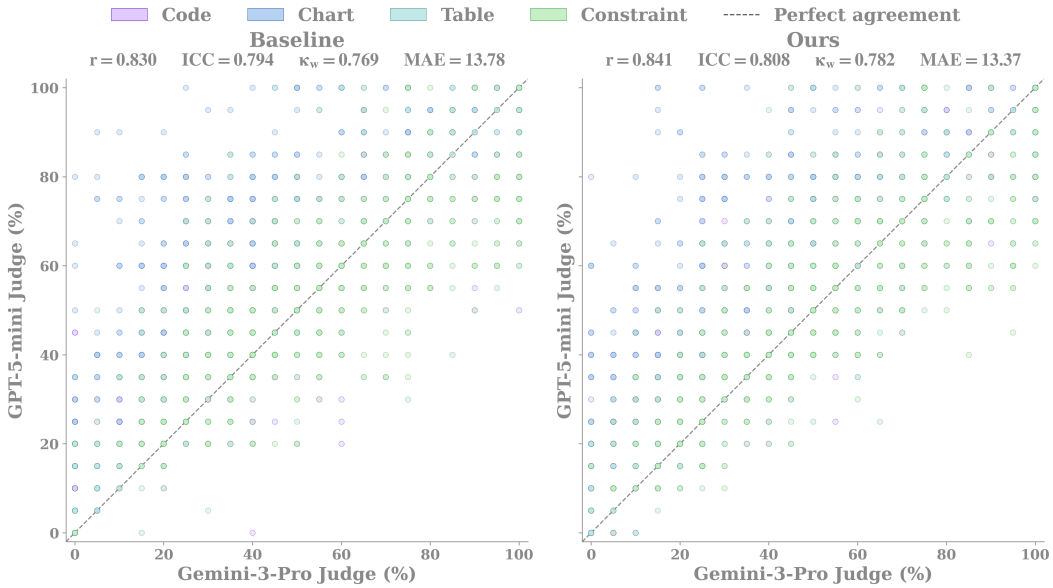


Figure 12: **Interrater Agreement.** We calculate interrater agreement between two LLM- and MLLM- judges. Results suggest close alignment between two judges, revealing the effectiveness of **CoCoEvolve@Eval** for cross-representation evaluation.

The three families are intentionally distinguished by symbol to make their scope, *i.e.*, intra-component, inter-component training, and evaluation, immediately clear at each point of use. Their detailed configurations are summarized in Tab. 6.

E.2 Computation Overhead

We summarize the computation overhead of **CoCoEvolve** across main training configurations and evaluation in Tab. 5, covering GPU resources, API cost, and wall-clock time.

Training. All training configurations run on H100 (96GB) GPUs with no API cost, as **CoCoEvolve** relies exclusively on consistency-driven rewards computed via deterministic sandbox execution and lightweight embedding models during training. Compared to the Baseline, **CoCoEvolve@Train** ($\mathcal{M}$ disabled) reduces wall-clock time, owing to the more efficient co-evolution objective as compared to sequential cross-representation learning. Enabling $\mathcal{M}$ further reduces training time to 71h (no annealing) and 72h (with annealing), as teacher-guided grounding accelerates early-stage policy convergence and reduces the number of low-quality rollouts that would otherwise contribute noisy reward signals.

Evaluation. All evaluation configurations similarly run on H100 (96GB) GPUs. The API cost decreases substantially across methods: baseline evaluation incurs $0.8637 per sample due to sequential generation and rollouts, while **CoCoEvolve** reduces the cost through test-time consistency-driven co-optimization (§3.3) by selecting the best candidates via annotation-free consistency signals. Wall-clock time follows the same trend, demonstrating that **CoCoEvolve** serves as an effective annotation-free optimization approach with reduced API cost and wall-clock time at test time.

F CoCoEvolve@Eval: Evaluation Suite for Multidimensional Assessment

Given the limitations of existing LLM-as-Judge approaches (§3.5), we propose **CoCoEvolve@Eval**, a systematic evaluation suite consisting of three complementary evaluation approaches across 27

metrics, covering all six cross-representation tasks in the representation cycle (§3.1): *rule-based judge* (§F.1), *LLM-as-Judge* (§F.2), and *MLLM-as-Judge* (§F.3).

F.1 Rule-as-Judge Evaluation

F.1.1 Rule-as-Judge Chart Evaluation

To achieve comprehensive chart evaluation, our rule-as-judge chart evaluation assesses four complementary dimensions: structural accuracy F_{ssim} (Eq. 20), semantic accuracy F_{clip} (Eq. 21), perceptual accuracy F_{dino} (Eq. 22), and textual accuracy F_{ocr} (Eq. 23). Concretely, given a model-predicted chart image $\hat{v}$ and a reference chart image v, each metric targets a distinct dimension of chart quality:

Structural Accuracy. F_{ssim} (Eq. 20) measures pixel-level structural similarity between $\hat{v}$ and v via the structural similarity index (Wang et al., 2004):

$$F_{ssim}(\hat{v}, v) = \texttt{SSIM}(\hat{v}, v) \in [0, 1] \tag{20}$$

where F_{ssim} captures local luminance, contrast, and structural patterns that reflect the layout and compositional fidelity of predicted charts.

Semantic Accuracy. F_{clip} (Eq. 21) measures semantic visual similarity between $\hat{v}$ and v via normalized CLIP (Radford et al., 2021) embeddings:

$$F_{clip}(\hat{v}, v) = \frac{\cos(e_{clip}(\hat{v}),\ e_{clip}(v)) + 1}{2} \in [0, 1] \tag{21}$$

where $e_{clip}(\cdot)$ denotes the visual embedding functions of CLIP (Radford et al., 2021), and thus F_{clip} captures high-level semantic alignment between the predicted and reference charts.

Perceptual Accuracy. F_{dino} (Eq. 22) measures fine-grained perceptual similarity between $\hat{v}$ and v via normalized DINOv2 (Oquab et al., 2024) embeddings:

$$F_{dino}(\hat{v}, v) = \frac{\cos(e_{dino}(\hat{v}),\ e_{dino}(v)) + 1}{2} \in [0, 1] \tag{22}$$

where $e_{dino}(\cdot)$ denotes the visual embedding functions of DINOv2 (Oquab et al., 2024), and thus F_{dino} captures spatial and feature-level visual fidelity beyond semantic-level alignment.

Textual Accuracy. F_{ocr} (Eq. 23) measures text-level fidelity between $\hat{v}$ and v by calculating token-level F_1 over OCR-extracted text via EasyOCR (JaidedAI, 2020):

$$F_{ocr}(\hat{v}, v) = F_1(\texttt{OCR}(\hat{v}),\ \texttt{OCR}(v)) \in [0, 1] \tag{23}$$

where F_{ocr} captures the textual accuracy of visual elements, such as axis labels, legends, titles, data annotations, etc.

Final Score. Based on these four complementary dimensions, the final rule-as-judge chart evaluation score F_{chart} is the weighted combination of four dimensions:

$$F_{chart}(\hat{v}, v) = \sum_{m} \gamma_m \cdot F_m(\hat{v}, v) \tag{24}$$

where $m \in \{clip, ssim, ocr, dino\}$, with $\gamma_m \in \{\gamma_{clip}, \gamma_{ssim}, \gamma_{ocr}, \gamma_{dino}\}$ as evaluation weighting coefficients. Our detailed configurations are summarized in §E.1 and Tab. 6.

F.1.2 Rule-as-Judge Code Evaluation

For more comprehensive code evaluation, our rule-as-judge code evaluation assesses six complementary dimensions: code executability F_{exec} (Eq. 25), code quality $F_{codebleu}$ (Eq. 26), structural accuracy F_{ast} (Eq. 27), lexical accuracy F_{cosine} (Eq. 28), contextual accuracy $F_{codebert}$, and semantic accuracy $F_{unixcoder}$. Concretely, given a model-predicted code $\hat{c}$ and a reference code c, each metric targets a distinct dimension of code quality:

Code Executability. F_{exec} measures whether the predicted code executes successfully via the deterministic sandbox executor h (Eq. 1):

$$F_{exec}(\hat{c}) = \mathbb{1}[h(\hat{c})] \in \{0, 1\} \tag{25}$$

We report F_{exec} as a standalone execution success rate metric to complement the final code evaluation score F_{code} (Eq. 29).

Code Quality. $F_{codebleu}$ measures overall code similarity via CodeBLEU (Ren et al., 2020), computed as a weighted combination of four complementary dimensions:

$$F_{codebleu}(\hat{c}, c) = \sum_{m} \gamma_m \cdot F_m(\hat{c}, c) \tag{26}$$

where $m \in \{ngram, wngram, syntax, dataflow\}$, with F_{ngram} as n-gram match, F_{wngram} as weighted n-gram match, F_{syntax} as syntax tree match, and $F_{dataflow}$ as data-flow graph match.

Structural Accuracy. F_{ast} measures structural similarity between $\hat{c}$ and c via Abstract Syntax Tree (AST) sequence matching:

$$\begin{aligned} F_{ast}(\hat{c}, c) &= \texttt{SequenceMatcher}(\texttt{AST}(\hat{c}), \texttt{AST}(c)) \\ &\in [0, 1] \end{aligned} \tag{27}$$

where $\texttt{AST}(\cdot)$ denotes the depth-first node label sequence of the parsed syntax tree, and `SequenceMatcher` computes the longest common subsequence (LCS) ratio.

Lexical Accuracy. F_{cosine} measures token-level lexical similarity between $\hat{c}$ and c via weighted cosine similarity over code tokens:

$$F_{cosine}(\hat{c}, c) = \frac{\mathbf{w}(\hat{c}) \cdot \mathbf{w}(c)}{\|\mathbf{w}(\hat{c})\| \cdot \|\mathbf{w}(c)\|} \in [0, 1] \tag{28}$$

where $\mathbf{w}(\cdot)$ denotes the TF-IDF (Salton & Buckley, 1988) weighted token vector over the vocabulary of identifier, number, and string tokens extracted via code tokenization.

Contextual Accuracy. $F_{codebert}$ measures contextual similarity between $\hat{c}$ and c via CodeBERT (Feng et al., 2020), capturing token-level contextual code representations by following the normalized embedding similarity formulation of $\texttt{sim}_c$ (Eq. 4).

Semantic Accuracy. $F_{unixcoder}$ measures semantic similarity between $\hat{c}$ and c via the unified cross-modal encoder of UniXcoder (Guo et al., 2022), capturing deeper code-to-code semantic correspondence by following the normalized embedding similarity formulation of $\texttt{sim}_c$ (Eq. 4).

Final Score. The final Rule-as-Judge code evaluation score F_{code} is the weighted combination across five dimensions:

$$F_{code}(\hat{c}, c) = \sum_{m} \gamma_m \cdot F_m(\hat{c}, c) \tag{29}$$

where $m \in \{codebleu, ast, cosine, codebert, unixcoder\}$, with F_{exec} (Eq. 25) reported separately as a standalone executability score (Tab. 1). Our detailed configurations are summarized in §E.1 and Tab. 6.

F.1.3 Rule-as-Judge Table Evaluation

Our rule-as-judge table evaluation assesses two complementary dimensions: schema accuracy and value accuracy. Concretely, given a model-predicted table $\hat{t}$ and a reference table t, each metric targets a distinct dimension of table quality:

Schema Accuracy. F_{schema} measures column-level F1 between $\hat{t}$ and t, directly instantiating the schema accuracy formulation of $\mathbf{F}_t$ (Eq. 12).

Value Accuracy. F_{value} measures cell-level F1 over all $(row, column, value)$ triples between $\hat{t}$ and t, directly instantiating the value accuracy formulation of $\mathbf{F}_t$ (Eq. 12).

Final Score. The final rule-as-judge table evaluation score F_{table} is the weighted combination across two dimensions:

$$F_{table}(\hat{t}, t) = \sum_m \gamma_m \cdot F_m(\hat{t}, t) \tag{30}$$

where $m \in \{schema, value\}$. Our detailed configurations are summarized in §E.1 and Tab. 6.

F.1.4 Rule-as-Judge Constraint Evaluation

Our rule-as-judge constraint evaluation assesses two complementary dimensions: semantic accuracy and lexical accuracy. Given a model-predicted constraint $\hat{s}$ and a reference constraint s_{gt}, each metric targets a distinct dimension of constraint quality:

Semantic Accuracy. $F_{ssem} = \mathbf{F}_s$ measures semantic similarity between $\hat{s}$ and s_{gt} via sentence-level cosine similarity, directly instantiating the formulation of $\mathbf{F}_s$ (Eq. 11).

Lexical Accuracy. F_{rouge} measures lexical similarity between $\hat{s}$ and s_{gt} via ROUGE-L (Lin, 2004):

$$F_{rouge}(\hat{s}, s_{gt}) = F_{LCS}(\hat{s}, s_{gt}) \in [0, 1] \tag{31}$$

where F_{LCS} denotes the F1 score computed over the longest common subsequence of tokens between the predicted and reference constraints.

Final Score. The final rule-as-judge constraint evaluation score $F_{constraint}$ is the weighted combination across two dimensions:

$$F_{constraint}(\hat{s}, s_{gt}) = \sum_m \gamma_m \cdot F_m(\hat{s}, s_{gt}) \tag{32}$$

where $m \in \{ssem, rouge\}$. Our detailed configurations are summarized in §E.1 and Tab. 6.

F.2 LLM-as-Judge Evaluation

Existing LLM-as-judge approaches for code evaluation collapse complex semantic and structural fidelity into a single and/or wide-range score, which LLMs are known to judge inconsistently and unreliably (§3.5). To address these limitations, our **CoCoEvolve@Eval** proposes LLM-as-judge evaluation that decomposes code quality into five fine-grained, independently scored dimensions, each targeting a distinct and non-overlapping dimension of visualization code quality (Fig. 14). Concretely, given the ground-truth code c and predicted code $\hat{c}$, the judge reasons about the expected rendered output via static code analysis, scoring each dimension on a 0–5 integer scale, and then normalizing to $[0, 1]$.

1. **Data Correctness**: correctness of underlying data values, categories, transformations, and their correspondence to the intended chart structure.
2. **Chart Type Accuracy**: correctness of chart type, subtype, dimensionality, and geometry (*e.g.*, grouped vs. stacked bars, single vs. multiple lines).
3. **Structural Fidelity**: correctness of layout, subplot configuration, axes, labels, titles, annotations, scales, ticks, legends, and grid configuration.
4. **Visual Accuracy**: correctness of data-to-visual encoding mappings, including positions, heights, trends, grouping, stacking, and category alignment.
5. **Style Accuracy**: correctness of colors, colormaps, marker types, sizes, transparency, line widths, fills, and other aesthetic details of each chart element.

The final LLM-as-Judge code evaluation score J_{code} is the weighted combination across five dimensions:

$$J_{code}(\hat{c}, c) = \sum_m \gamma_m \cdot J_m(\hat{c}, c) \tag{33}$$

where $m \in \{Cdata, Ctype, Cstruct, Cvisual, Cstyle\}$. Our detailed configurations are summarized in §E.1 and Tab. 6.

F.3 MLLM-AS-JUDGE EVALUATION

Similar to LLM-as-judge, existing MLLM-as-judge approaches collapse complex visual and structural fidelity into a single wide-range score, which MLLMs are known to judge unreliably and inconsistently (§3.5).For more reliable MLLM judgments, our **CoCoEvolve@Eval** decomposes each evaluation into five fine-grained, independently scored dimensions, each targeting a distinct and non-overlapping aspect of model prediction quality, scoring each dimension on a 0–5 integer scale normalized to $[0, 100]$. **CoCoEvolve@Eval** includes three MLLM-as-judge evaluators: chart evaluation (§F.3.1), table evaluation (§F.3.2), and constraint evaluation (§F.3.3), respectively.

F.3.1 MLLM-AS-JUDGE CHART EVALUATION

Given the ground-truth chart image v and predicted chart image $\hat{v}$, the judge directly compares the two rendered images across five dimensions, as detailed in Fig. 13:

1. **Chart Type Accuracy**: correctness of chart type and subtype, including exact match of chart geometry and dimensionality (e.g., grouped vs. stacked bars, pie vs. donut).
2. **Data Accuracy**: accuracy of data values, proportions, magnitudes, and precisions as visually perceived in the predicted chart.
3. **Text Accuracy**: correctness of all text elements, including titles, axis labels, tick values, legend entries, and annotations, together with their positions and formatting.
4. **Style Accuracy**: correctness of visual styles of chart elements, including colors, colormaps, markers, line styles, fills, and their assignment to the correct data series.
5. **Visual Accuracy**: correctness of overall visual layout, spatial relationships, and structural details, including figure proportions, subplot arrangement, spacing, stacking order, category ordering, trends, and decorative details such as gridlines, error bars, and reference lines.

The final MLLM-as-judge chart evaluation score J_{chart} is the weighted combination across five dimensions:

$$J_{chart}(\hat{v}, v) = \sum_{m} \gamma_m \cdot J_m(\hat{v}, v) \tag{34}$$

where $m \in \{Vtype, Vdata, Vtext, Vstyle, Vvisual\}$. Our detailed configurations are summarized in §E.1 and Tab. 6.

F.3.2 MLLM-AS-JUDGE TABLE EVALUATION

Given the ground-truth table t, predicted table $\hat{t}$, ground-truth code c, and ground-truth chart image v as reference context, the judge evaluates the predicted table across five dimensions, as detailed in Fig. 15:

1. **Schema Accuracy**: correctness of column headers and table structure, penalizing missing, extra, or incorrectly named columns.
2. **Data Accuracy**: correctness of all cell values compared to ground truth, penalizing hallucinated, incorrect, or miscomputed values.
3. **Data Coverage**: completeness of extracted data relative to ground truth, penalizing missing rows or partially captured data.
4. **Row Alignment**: correctness of row ordering and alignment of values across all columns.
5. **Format Correctness**: validity of table structure, including correct column-keyed format, string-typed values, equal-length columns, and correct representation of missing values.

The final MLLM-as-judge table evaluation score J_{table} is the weighted combination across five dimensions:

$$J_{table}(\hat{t}, t) = \sum_{m} \gamma_m \cdot J_m(\hat{t}, t) \tag{35}$$

where $m \in \{Tschema, Tdata, Tcoverage, Trow, Tformat\}$. Our detailed configurations are summarized in §E.1 and Tab. 6.

F.3.3 MLLM-AS-JUDGE CONSTRAINT EVALUATION

Given the ground-truth constraint s_{gt}, predicted constraint $\hat{s}$, ground-truth code c, and ground-truth chart image v as reference context, the judge evaluates the predicted constraint across five dimensions, as detailed in Fig. 16:

1. **Visual Fidelity**: correctness of all key visual contents and data values described in $\hat{s}$ relative to s_{gt}, penalizing missing, incorrect, or hallucinated contents and values.
2. **Visual Completeness**: completeness of all important visual elements described in $\hat{s}$, including axes, labels, legends, trends, categories, and colors, penalizing missing key components needed for chart reconstruction.
3. **Structural Clarity**: quality of organization of $\hat{s}$ in support of chart reproduction, including clarity of relationships between visual elements.
4. **Trend and Relationship Accuracy**: correctness of described trends, comparisons, and relationships between data elements, penalizing incorrect interpretations such as wrong directional trends.
5. **Style Accuracy**: correctness and completeness of described plotting colors and styles, penalizing incorrect or missing style descriptions.

The final MLLM-as-judge constraint evaluation score $J_{constraint}$ is the weighted combination across five dimensions:

$$J_{constraint}(\hat{s}, s_{gt}) = \sum_{m} \gamma_m \cdot J_m(\hat{s}, s_{gt}) \tag{36}$$

where $m \in \{Sfidelity, Scompleteness, Sclarity, Strend, Sstyle\}$. Our detailed configurations are summarized in §E.1 and Tab. 6.

G ANALYSIS ON CROSS-REPRESENTATION FAILURES

G.1 QUANTITATIVE ANALYSIS ON CROSS-REPRESENTATION FAILURES

Leveraging five-dimensional evaluation for each representation (§3.5), Fig. 10 presents the failure distribution of cross-representation understanding for the baseline, **CoCoEvolve@Train**, and **CoCoEvolve@Train+@Test**. As the results show, **CoCoEvolve** substantially outperforms the non-finetuned base model, achieving an overall improvement of up tp $\Delta =\uparrow 35.10\%$. This gain is primarily accompanied by notable reductions in data-related errors across different representations, while the remaining categories of failures are also consistently alleviated. This finding underscores the importance of data accuracy and information completeness for cross-representation understanding success. Furthermore, applying **CoCoEvolve@Test** leads to additional reductions across all five failure dimensions of each representation. Its resulting correctness improvement of up to $\Delta =\uparrow 41.38\%$ further validates the effectiveness of **CoCoEvolve** in optimizing model cross-representation understanding during both training and testing.

G.2 QUALITATIVE ANALYSIS ON CROSS-REPRESENTATION FAILURES

To qualitatively analyze cross-representation failures of different models, we manually inspect model outputs and evaluate how each model performs. Overall, larger models showcase stronger cross-representation understanding capabilities than smaller-scale models. Through detailed examination, Fig. 18 illustrates a relatively simple case in which chart elements are explicitly labeled, mitigating the negative impact of inaccurate data extraction. However, `Claude-4.6-Sonnet` still exhibits style-related errors, failing to preserve the correct value ranges and color encodings. On the other hand, Fig. 17 shows the cross-representation outputs of `Qwen3-4B-VL`, revealing that its primary failures stem from data recognition and extraction errors. These inaccuracies also exist across representations and modalities, leading to inconsistent outputs. Its failures in color encoding and cross-representation consistency further highlight the importance of systematic evaluation signals for both accurately assessing cross-representation understanding and guiding model learning and optimization.

Table 4: **Notation and Symbol Reference.** We summarize core symbols and notations used in this paper, grouped by functional role. The three weighting coefficient families are intentionally distinct: ω_* denotes sub-weights within a reward component, λ_* denotes weights combining reward components during training, and γ_* denotes weights combining metrics during evaluation. LLM-as-judge and MLLM-as-judge dimension scores use prefixes C, V, T, S to disambiguate dimensions sharing the same name across evaluators.

Notation	Definition
	Representation Spaces & Data
$\mathcal{V}, \mathcal{T}, \mathcal{C}, \mathcal{S}$	Spaces of chart images, tabular data, visualization code, and descriptive constraints
v, t, c, s	A chart image, table, visualization code, and constraint
$\hat{v}, \hat{t}, \hat{c}, \hat{s}$	Model-predicted chart image, table, code, and constraint
s_{gt}, t_{gt}	Ground-truth constraint and table annotations
	Models, Policies & Task Mappings
M_θ, M_ψ	Perform $\mathcal{V} \to \mathcal{C}$ and $\mathcal{T} \to \mathcal{C}$, respectively, with constraints
π_θ, π_ψ	Policies parameterizing M_θ and M_ψ during RL training
$\pi_\theta^{ref}, \pi_\psi^{ref}$	Frozen reference policies for KL regularization
f_θ, g_ψ, h	Task mappings instantiated by M_θ, M_ψ, and sandbox executer, respectively
	Teacher Module & Training Configuration
$\mathcal{M}$	Optional teacher module providing ground-truth conditioning context
$\alpha, \alpha_{start}, \alpha_{end}$	Teacher-guidance fraction and its initial/final values under linear annealing
$\phi(\cdot)$	Sentence embedding function via SentenceBERT
N_{train}, N_{epoch}, lr	Total training steps, epochs, and initial learning rate for `AdamW`
	Rollouts & Sampling
$o_\theta^{(i)}, o_\psi^{(i,k)}$	i-th rollout $(\hat{s}^{(i)}, \hat{t}^{(i)}, \hat{c}_\theta^{(i)})$ from π_θ; k-th rollout from π_ψ conditioned on $o_\theta^{(i)}$
K_θ, K_ψ	Number of rollouts sampled per chart from π_θ; per π_θ rollout from π_ψ
B	Global batch size
τ	Token position index within a rollout sequence
$\rho_{\theta,\tau}^{(i)}, \rho_{\psi,\tau}^{(i,k)}$	Per-token importance ratios for π_θ and π_ψ at token position τ
ϵ, β, δ	Clipping threshold, KL penalty coefficient, and numerical stabilizer
	Rewards & Advantage Estimation
$\mathbf{F}_c, \mathbf{F}_v, \mathbf{F}_f$	Code consistency, visual consistency, and format reward components
$\mathbf{F}_s, \mathbf{F}_t$	Constraint and table grounding rewards (teacher mode only)
$\mathtt{sim}_c(c_a, c_b)$	Normalized code embedding cosine similarity between c_a and c_b
e_a, e_b, e_{base}	Code embeddings of c_a, c_b, and the baseline anchor in $\mathtt{sim}_c$
$e_{clip}(\cdot), e_{dino}(\cdot)$	Visual embedding functions of CLIP and DINOv2
$R_\theta, R_\psi, R_\theta^+$	Hierarchical rewards for π_θ, π_ψ, and π_θ under teacher-guided mode
r_θ, r_ψ	Paired reward functionals depending on both models' outputs
$\bar{r}_\theta^{(i)}$	π_θ reward marginalized over K_ψ children rollouts
$A_\theta^{(i)}, A_\psi^{(i,k)}$	Normalized advantages for π_θ rollout i and π_ψ rollout (i,k)
$\mu_\theta, \sigma_\theta, \mu_\psi^{(i)}, \sigma_\psi^{(i)}$	Mean and std for π_θ and π_ψ advantage normalization
	Rule-as-Judge Evaluation Metrics
$F_{clip}, F_{ssim}, F_{ocr}, F_{dino}$	Chart evaluation: semantic, structural, textual, and perceptual accuracy
F_{exec}	Code executability: binary sandbox execution success
$F_{codebleu}, F_{ast}, F_{cosine}, F_{codebert}, F_{unixcoder}$	Code evaluation: quality, structural, lexical, contextual, and semantic accuracy
$F_{ngram}, F_{wngram}, F_{syntax}, F_{dataflow}$	Four sub-dimensions of CodeBLEU comprising $F_{codebleu}$
F_{schema}, F_{value}	Table evaluation: column-level schema F1 and cell-level value F1
$F_{ssem}, F_{rouge}, F_{LCS}$	Constraint evaluation: semantic accuracy, ROUGE-L, and LCS F1
$F_{chart}, F_{code}, F_{table}, F_{constraint}$	Final rule-as-judge score for chart, code, table, and constraint evaluation

Notation	Definition
LLM-as-Judge Evaluation Metrics (prefix C: code)	
$J_{Cdata}, J_{Ctype}, J_{Cstruct}, J_{Cvisual}, J_{Cstyle}$	LLM-as-judge code dimensions: data correctness, chart type, structural fidelity, visual and style accuracy
J_{code}	Final LLM-as-judge code evaluation score
MLLM-as-Judge Evaluation Metrics (prefixes V: chart, T: table, S: constraint)	
$J_{Vtype}, J_{Vdata}, J_{Vtext}, J_{Vstyle}, J_{Vvisual}$	MLLM-as-judge chart dimensions: type, data, text, style, and visual accuracy
$J_{Tschema}, J_{Tdata}, J_{Tcoverage}, J_{Trow}, J_{Tformat}$	MLLM-as-judge table dimensions: schema, data, coverage, row alignment, and format correctness
$J_{Sfidelity}, J_{Scompleteness}, J_{Sclarity}, J_{Strend}, J_{Sstyle}$	MLLM-as-judge constraint dimensions: fidelity, completeness, clarity, trend accuracy, and style
$J_{chart}, J_{table}, J_{constraint}$	Final MLLM-as-judge score for chart, table, and constraint evaluation
Weighting Coefficient Families	
$\omega_*^{(\pi)}$	Sub-weights within a reward component
$\lambda_*^{(\pi)}$	Weighting coefficients combining reward components during training
γ_*	Weighting coefficients combining metrics during evaluation
Dataset & Filtering Thresholds	
$l_{input}, l_{code}, l_{table}, l_{constraint}, l_{timeout}$	Token thresholds for input, code, table, and constraint; sandbox execution timeout

Table 5: **Computation Overhead.** We summarize GPU resources, API cost, and wall-clock time for main training configurations and evaluation. API cost is calculated for per sample averaged across all evaluation benchmarks and decreases with **CoCoEvolve@Test**. Wall-clock time for training is calculated for per full training run; for evaluation it is calculated for per sample averaged across all benchmarks. "–" indicates no API cost is incurred during training.

Method	GPU	API Cost	Time
Training			
Baseline	4× H100 96GB	–	86h / run
CoCoEvolve ($\mathcal{M}$ disabled)	4× H100 96GB	–	77h / run
CoCoEvolve ($\mathcal{M}$ enabled, no annealing)	4× H100 96GB	–	71h / run
CoCoEvolve ($\mathcal{M}$ enabled, with annealing)	4× H100 96GB	–	72h / run
Evaluation			
Baseline	4× H100 96GB	\$0.8637 / sample	128.72s / sample
CoCoEvolve	4× H100 96GB	\$0.5674 / sample	77.63s / sample
CoCoEvolve (w/ **CoCoEvolve@Test**)	4× H100 96GB	\$0.2382 / sample	71.88s / sample

Table 6: **Experiment Configuration.** We summarize our core hyperparameter settings across different experimental stages. Parameters specific to teacher mode ($\lambda_s^{(\theta)}, \lambda_t^{(\theta)}, \omega_{schema}^{(\theta)}, \omega_{value}^{(\theta)}$) apply to M_θ only and are marked "–" for M_ψ. Ablation study rows list only parameters that differ from the corresponding main training setting. In our evaluation, we set the evaluation weighting coefficients γ_* to be uniform within each score.

Parameter	M_θ	M_ψ
Dataset Construction		
l_{input}	8,192	8,192
l_{code}	8,192	8,192
l_{table}	8,192	8,192
$l_{constraint}$	8,192	8,192
$l_{timeout}$	60	60
Main Training ($\mathcal{M}$ Disabled)		
N_{epoch}	2	2
N_{train}	200	200
B	8	8
K_θ, K_ψ	4, 4	4, 4
lr	1×10^{-6}	1×10^{-6}
α	0	0
$\lambda_f^{(\pi)}$	0.1	0.1
$\lambda_c^{(\pi)}$	0.05	0.2
λ_π	0.7	0.7
$\lambda_v^{(\pi)}$	0.05	0.2
$\omega_e^{(\pi)}$	0.9	0.7
$\omega_s^{(\pi)}$	0.1	0.3
$\omega_{clip}^{(\pi)}$	0.25	0.25
$\omega_{ssim}^{(\pi)}$	0.25	0.25
$\omega_{ocr}^{(\pi)}$	0.25	0.25
$\omega_{dino}^{(\pi)}$	0.25	0.25
Ablation Study: Teacher Module ($\mathcal{M}$ Enabled, No Annealing)		
α_{start}	1.0	1.0
α_{end}	1.0	1.0
$\lambda_s^{(\theta)}$	0.2	–
$\lambda_t^{(\theta)}$	0.3	–
$\omega_{schema}^{(\theta)}$	0.5	–
$\omega_{value}^{(\theta)}$	0.5	–
Ablation Study: Teacher Module ($\mathcal{M}$ Enabled, With Annealing)		
α_{start}	1.0	1.0
α_{end}	0.0	0.0
Ablation Study: Consistency Reward Weighting		
$\lambda_c^{(\pi)}$	0.2	0.2
$\lambda_v^{(\pi)}$	0.2	0.2
λ_π	0.5	0.5
$\omega_e^{(\pi)}$	0.5	0.5
$\omega_s^{(\pi)}$	0.5	0.5
Evaluation		
γ_* in F_{chart}	0.25	0.25
γ_* in F_{code}	0.20	0.20
γ_* in F_{bleu}	0.25	0.25
γ_* in F_{table}	0.50	0.50
γ_* in $F_{constraint}$	0.50	0.50
γ_* in J_{code}	0.20	0.20
γ_* in J_{chart}	0.20	0.20
γ_* in J_{table}	0.20	0.20
γ_* in $J_{constraint}$	0.20	0.20

MLLM-as-Judge for Chart Evaluation

```
You are an expert data visualization evaluator specialized in evaluating predicted chart images agaist ground-truth charts.

## **Task**
You will be given two chart images:
* **Reference Chart**: the original ground-truth chart
* **Predicted Chart**: a chart reproduced by an AI model from Python code
Your task is to carefully compare the two charts and score the **Predicted Chart** across 5 dimensions.

## **Input**
**Reference Chart**: <image>
**Predicted Chart**: <image>

## **Evaluation Dimensions (score 0-5 each, the higher the better)**
1. chart_type_accuracy:
   * Correct chart type(s) (bar, line, pie, scatter, area, histogram, heatmap, etc.)
   * Correct subtype and combination (grouped vs. stacked bars, single vs. multiple lines, mixed types such as bar+line,
types across subplots)
   * All chart types in the reference are present, with no extra or substituted types
2. data_accuracy:
   * Correct data values, proportions, magnitudes, and precisions
   * Correct relative ordering, peak/valley positions, and overall shape across all data points and series
   * Penalize missing, incorrect, or fabricated data points and series
3. text_accuracy:
   * Correct content of all text elements (titles, axis labels, tick values, legend entries, value labels, annotations)
   * Correct formats, precisions, and tick intervals
   * Correct positions and assignments (e.g., axis labels not swapped, legend entries matched to the right series)
   * Penalize missing, truncated, wrong, or mispositioned text
4. style_accuracy:
   * Correct colors, line styles, markers, fills, opacity, and widths of chart elements (bars, lines, markers, segments, areas)
   * Styles correctly assigned to the corresponding data series, with no swaps between series
   * Penalize mismatched, swapped, or invented styles
5. visual_accuracy:
   * **Layout**: figure size proportions, subplot arrangement, spacing, margins, padding
   * **Spatial Relations**: overlapping vs. non-overlapping bars, stacking order, grouped vs. dodged positioning, side-by-
side vs. layered elements
   * **Semantic Relations**: relative sizes, heights, trends, relationships, colors, etc.
   * **Trends & Shapes**: curve shapes, peak/valley positions, crossing points, slopes for line/area charts
   * **Proportions**: relative segment sizes in pie/donut charts, relative bar heights, relative areas
   * **Ordering**: sorted vs. unsorted bars, clockwise/counterclockwise pie segments, category ordering, legend ordering
   * **Decorative Details**: gridlines, error bars, reference lines, markers, line styles, bar widths, opacity, hatching

## **Scoring Rules**
* Each score must be an integer from 0 to 5
   * 5: Perfect or near-perfect match
   * 3-4: Mostly correct, with minor errors or omissions
   * 1-2: Partially correct, but significant errors or missing elements
   * 0: Completely wrong or absent
* Be strict: partial correctness should not get full marks
* Check and compare carefully
* Penalize hallucinated or invented visual elements heavily

## **Important Requirements**
* Output MUST be valid JSON only, following the **Output Format** below (no markdown, no extra text)
* Keep each "reason" field concise (max 1-2 sentences)

## **Output Format (STRICT JSON ONLY)**
Return a single JSON object in this exact structure:
{
   "chart_type_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "data_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "text_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "style_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "visual_accuracy": {"score": 0-5, "reason": "brief, concrete"}
}
```

Figure 13: **MLLM-as-Judge for Chart Evaluation.** Our MLLM-as-judge for chart evaluation assesses predicted chart against ground-truth chart across five fine-grained dimensions, including chart type accuracy, data accuracy, text accuracy, style accuracy, and visual accuracy (§F.3.1). Each dimension is scored on a 0–5 integer scale with detailed instructions from reasoning to scoring, addressing the key limitations in existing MLLM-as-judge approaches (§3.5).

LLM-as-Judge for Code Evaluation

You are an expert data visualization evaluator specialized in evaluating whether a predicted Python visualization code correctly reproduces visually equivalent chart as the ground-truth Python code.

Task
Your task is to evaluate whether the **Predicted Code** correctly reproduce a visually equivalent chart as the **Ground-Truth Code** in terms of data, structure, and visual design.
You MUST judge based on the expected FINAL RENDERED OUTPUT implied by the code, as it would appear to a human viewer, using static code reasoning only.

Input
Ground-Truth Code
```python
{GT_CODE}
```
Predicted Code
```python
{PRED_CODE}
```

Evaluation Goal
The predicted code should, if executed, reproduce a chart that is visually and structurally equivalent to the chart image generated by the ground-truth code, including:
* correct data values and data transformations
* correct chart types and subtypes
* correct axes, labels, legends, and layout
* correct grouping, ordering, stacking, and category mapping
* correct visual encoding (positions, sizes, colors, styles)

Evaluation Dimensions (score 0-5 each, the higher the better)
1. data_correctness:
* Correct underlying data values, categories, and structure
* Correct transformations (aggregation, filtering, sorting, normalization, scaling)
* No missing, incorrect, or hallucinated data
* Data schema aligns with the intended chart
* Judge whether the same quantitative information would be perceived by a viewer
2. chart_type_accuracy:
* Correct chart type (bar, line, scatter, histogram, heatmap, etc.)
* Correct subtype (different types in one chart or across subplots, grouped vs stacked bars, multiple lines vs single line, etc.)
* Correct dimensionality and geometry
3. structural_fidelity:
* Layout, subplot configuration, and figure size/aspect ratio
* Axes, labels, titles, annotations (text content should be meaningfully correct)
* Axis scales (linear/log), ranges/limits, ticks, and orientation
* Legend: presence, labels, ordering, and placement
* Grid: presence, axis (x/y/both), and basic style
* Correct grouping, ordering, and category alignment
4. visual_accuracy:
* Data correctly mapped to visual encodings (positions, heights, lengths, trends)
* Correct grouping, stacking, and hue/category mapping
* Correct ordering of categories or data series
* No mismatches between data and what is visually displayed
5. style_accuracy:
* Colors and colormaps (including consistency, mapping, and direction if applicable)
* Styles, marker types and sizes, alpha/transparency, widths, markers, etc., of each chart element (e.g., lines, scatters, etc.)
* Edge styles, fills, and other aesthetic details
* Judge the final rendered appearance, not parameter names

```
## **Scoring Rules**
* Each score must be an integer from 0 to 5
   * 5: Perfect or near-perfect match
   * 3-4: Mostly correct, with minor errors or omissions
   * 1-2: Partially correct, but significant errors or missing elements
   * 0: Completely wrong or absent
* Be strict: partial correctness should not get full marks
* Penalize any hallucinated or invented components heavily
* If data, chart type, or visual mapping is incorrect, scores should be heavily reduced
* Missing or incorrect legends, axes, or labels should significantly reduce structural scores
* If visual differences are clearly noticeable, do not assign scores above 4
* If fundamentally incorrect (wrong data or chart type), scores should be 0-2
* If the output cannot be reliably inferred from the code, reduce scores

## **Important Requirements**
* Do not give benefit of the doubt when information is missing or ambiguous
* Output MUST be valid JSON only, following the **Output Format** below (no markdown, no extra text)
* Keep each "reason" field concise (max 1-2 sentences)

## **Output Format (STRICT JSON ONLY)**
Return a single JSON object in this exact structure:
{
   "data_correctness": {"score": 0-5, "reason": "brief, concrete"},
   "chart_type_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "structural_fidelity": {"score": 0-5, "reason": "brief, concrete"},
   "visual_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "style_accuracy": {"score": 0-5, "reason": "brief, concrete"}
}
```

Figure 14: **LLM-as-Judge for Code Evaluation.** Our LLM-as-judge evaluates predicted visualization code against ground-truth code across five fine-grained dimensions, including data correctness, chart type accuracy, structural fidelity, visual accuracy, and style accuracy (§F.2). Each dimension is scored on a 0–5 integer scale with detailed instructions from reasoning to scoring, addressing the key limitations in existing LLM-as-judge approaches (§3.5).

MLLM-as-Judge for Table Evaluation

```
You are an expert data analyst specialized in evaluating structured data extraction from chart-generating Python
code.

## **Task**
Your task is to carefully compare **Predicted Table** against the **Ground-Truth Table** using the provided
**Python Code** and **Chart Image**, and then produce a structured, strictly formatted evaluation.

## **Inputs**
### **Python Code**
```python
{PYTHON_CODE}
```
Chart Image
{CHART_IMAGE}
Ground-Truth Table
{GT_TABLE}
Predicted Table
{MODEL_TABLE}

Evaluation Dimensions (score 0-5 each, the higher the better)
1. schema_accuracy:
   * Correctness of column headers and table structure
   * Penalize missing, extra, or incorrect column names
2. data_accuracy:
   * Correctness of all values compared to ground truth
   * Penalize hallucinated, incorrect, or miscomputed values
3. data_coverage:
   * Completeness of extracted data vs ground truth
   * Penalize missing rows or partially captured data
4. row_alignment:
   * Correct ordering and alignment of rows across all columns
5. format_correctness: Valid structure
   * Each key is a column header (string), with its corresponding value as a list of strings representing that column's
data values, in row order
   * All values must be strings
   * Equal-length columns
   * Missing values correctly represented as "-"

Scoring Rules
* Each score must be an integer from 0 to 5
   * 5: Perfect or near-perfect match
   * 3-4: Mostly correct, with minor errors or omissions
   * 1-2: Partially correct, but significant errors or missing elements
   * 0: Completely wrong or absent
* Be strict: partial correctness should not get full marks
* Use chart image + code + GT table for verification
* Penalize hallucinated or invented values heavily

Important Requirements
* Output MUST be valid JSON only, following the **Output Format** below (no markdown, no extra text)
* Keep each "reason" field concise (max 1-2 sentences)

Output Format (STRICT JSON ONLY)
Return a single JSON object in this exact structure:
{
   "schema_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "data_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "data_coverage": {"score": 0-5, "reason": "brief, concrete"},
   "row_alignment": {"score": 0-5, "reason": "brief, concrete"},
   "format_correctness": {"score": 0-5, "reason": "brief, concrete"}
}
```

Figure 15: **MLLM-as-Judge for Table Evaluation.** Our MLLM-as-judge for table evaluation assesses the extracted table against the reference table across five fine-grained dimensions, including schema accuracy, data accuracy, data coverage, row alignment, and format correctness (§F.3.2). Each dimension is scored on a 0–5 integer scale with detailed instructions from reasoning to scoring, addressing the key limitations in existing MLLM-as-judge approaches (§3.5).

MLLM-as-Judge for Constraint Evaluation

```
You are an expert visual data analyst specialized in evaluating descriptive summaries of chart images.

## **Task**
Your task is to evaluate how well a predicted descriptive summary captures the full visual content of a chart
image.

## Inputs
### **Python Code**
```python
{PYTHON_CODE}
```
Chart Image
{CHART_IMAGE}
Ground-Truth Summary
{GT_SUMMARY}
Predicted Summary
{MODEL_SUMMARY}

Evaluation Dimensions (score 0-5 each, the higher the better)
1. visual_fidelity:
   * Are all key visual contents and data values correctly described as **Ground-Truth Summary**?
   * Penalize missing, incorrect, or hallucinated contents and values
2. visual_completeness:
   * Are all important visual elements described? (e.g., axes, labels, legends, trends, categories, colors, etc. if
relevant)
   * Penalize missing key components needed for reconstruction
3. structural_clarity:
   * Is the description organized in a way that supports chart reproduction?
   * Are relationships between elements clearly explained?
4. trend_and_relationship_accuracy:
   * Are trends, comparisons, and relationships correctly described?
   * Penalize incorrect interpretations (e.g., wrong increasing/decreasing trends)
5. style_accuracy:
   * Are all plotting colors and styles clearly described?
   * Penalize incorrect and missing descriptions

Scoring Rules
* Each score must be an integer from 0 to 5
   * 5: Perfect or near-perfect match
   * 3-4: Mostly correct, with minor errors or omissions
   * 1-2: Partially correct, but significant errors or missing elements
   * 0: Completely wrong or absent
* Be strict: missing even one key value, label, relation, trend, or style, should reduce score significantly
* Use chart image + code + GT summary for verification
* Penalize hallucinated or invented details heavily

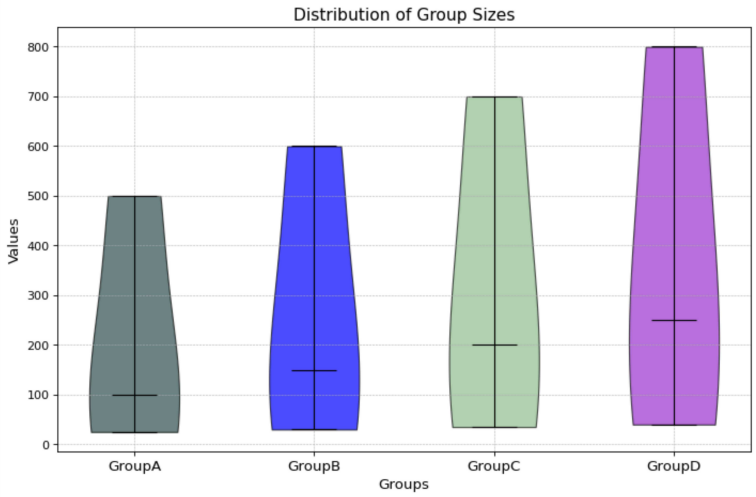
Important Requirements
* Output MUST be valid JSON only, following the **Output Format** below (no markdown, no extra text)
* Keep each "reason" field concise (max 1-2 sentences)

Output Format (STRICT JSON ONLY)
Return a single JSON object in this exact structure:
{
   "visual_fidelity": {"score": 0-5, "reason": "brief, concrete"},
   "visual_completeness": {"score": 0-5, "reason": "brief, concrete"},
   "structural_clarity": {"score": 0-5, "reason": "brief, concrete"},
   "trend_and_relationship_accuracy": {"score": 0-5, "reason": "brief, concrete"},
   "style_accuracy": {"score": 0-5, "reason": "brief, concrete"}
}
```

Figure 16: **MLLM-as-Judge for Constraint Evaluation.** Our MLLM-as-judge for constraint evaluation assesses the predicted constraint against the reference constraint across five fine-grained dimensions, including visual fidelity, visual completeness, structural clarity, trend & relation accuracy, and style accuracy (§F.3.3). Each dimension is scored on a 0–5 integer scale with detailed instructions from reasoning to scoring, addressing the key limitations in existing MLLM-as-judge approaches (§3.5).

Cross-Representation Failure 1

Original Chart



$\mathcal{V} \rightarrow \mathcal{T}$

Group	Min_Value	Q1	Median	Q3	Max_Value
GroupA	0	0	100	500	500
GroupB	0	0	150	600	600
GroupC	0	0	200	700	700
GroupD	0	0	250	800	800



Data extraction errors: Model-generated table shows substantial inaccuracies as compared to the original chart. For instance, the minimum value of all four groups are incorrectly extracted as 0, whereas it can be clearly seen that all four groups have their minimum value above 0, and GroupA < GroupB < GroupC < GroupD.

$\mathcal{V} \rightarrow \mathcal{C}$

```
import matplotlib.pyplot as plt
import numpy as np

# Data from the table
groups = ["GroupA", "GroupB", "GroupC", "GroupD"]
min_values = [0, 0, 0, 0]
q1_values = [0, 0, 0, 0]
medians = [100, 150, 200, 250]
q3_values = [500, 600, 700, 800]
max_values = [500, 600, 700, 800]

# Create the figure
fig, ax = plt.subplots(figsize=(10, 6))

# Create a dataset for the violin plot
# We will use the data to represent the distribution
# For simplicity, we create a dataset that reflects the provided quartiles and ranges
np.random.seed(42)  # For reproducibility

# Create data for each group
data = {
    "GroupA": np.random.normal(500, 100, 100),
    "GroupB": np.random.normal(600, 150, 100),
    "GroupC": np.random.normal(700, 175, 100),
    "GroupD": np.random.normal(800, 200, 100),
}

# Create the violin plot
violin_plot = ax.violinplot(
    [data[grp] for grp in groups],
    showmeans=False,
    showmedians=True
)

# Set the x-axis labels
ax.set_xticks([1, 2, 3, 4])
ax.set_xticklabels(groups)

# Set the y-axis range
ax.set_ylim(0, 800)
ax.set_xlabel("Groups")
ax.set_ylabel("Values")
ax.set_title("Distribution of Group Sizes")

# Customize the appearance
# Use different colors for each group
colors = ['#2C2C2C', '#1E90FF', '#98FB98', '#6A5ACD']
for i, patch in enumerate(violin_plot['bodies']):
    patch.set_facecolor(colors[i])
    patch.set_edgecolor('black')
    patch.set_alpha(0.3)

# Set the style to be clean
ax.grid(True, linestyle='--', linewidth=0.5, alpha=0.3)

# Adjust layout to prevent clipping
plt.tight_layout()

# Save the figure
plt.savefig("/workspace/generated_chart.png", dpi=300, bbox_inches="tight")

# Show the figure
plt.show()
```



1. Propagation of data extraction errors: The model-generated visualization code inherits the same failure patterns observed in the table, resulting in incorrect numerical values and incomplete data reconstruction.
2. Color encoding inconsistency: The visualization code fails to preserve the original color mapping, producing colors that differ substantially from those in the source visualization.

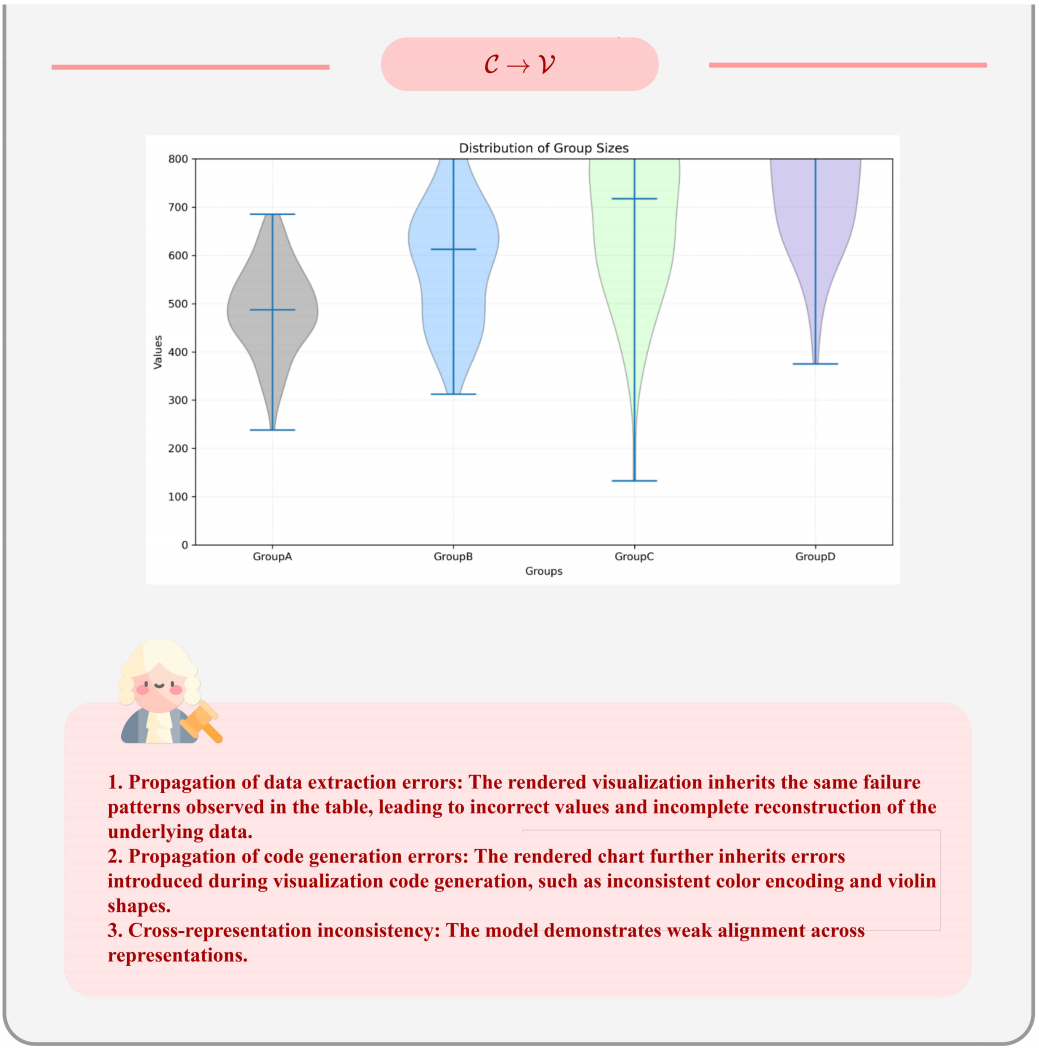
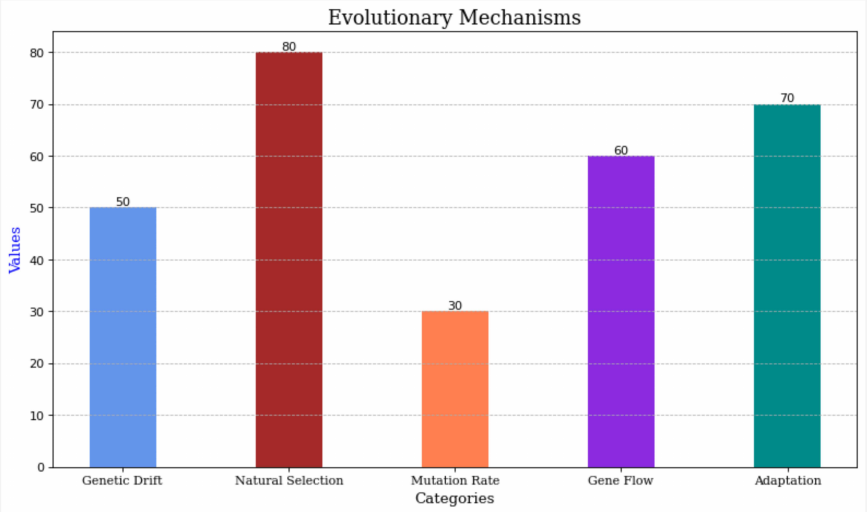


Figure 17: **Cross-Representation Understanding Failure Example of `Qwen3-4B-VL`.** A case study on `Qwen3-4B-VL` cross-representation understanding failures, illustrating how small-size models produce data recognition errors that accumulate and propagate across representations, resulting in inaccurate data extraction, incomplete information transfer, and significant cross-representation inconsistencies. Constraints are omitted for clarity.

Cross-Representation Failure

Original Chart



$\mathcal{V} \rightarrow \mathcal{T}$

Category	Value
Genetic Drift	50
Natural Selection	80
Mutation Rate	30
Gene Flow	60
Adaptation	70



When value labels are provided, the model is able generate more accurate table.

$\mathcal{V} \rightarrow \mathcal{C}$

```
import matplotlib.pyplot as plt

# Data
categories = ["Genetic Drift", "Natural Selection", "Mutation Rate", "Gene Flow", "Adaptation"]
values = [50, 80, 30, 60, 70]
colors = ["cornflowerblue", "firebrick", "coral", "mediumpurple", "teal"]

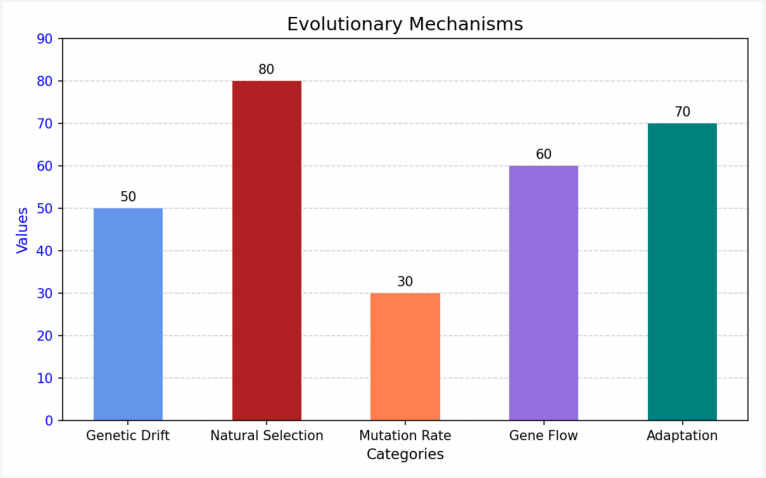
# Plot
fig, ax = plt.subplots(figsize=(8, 5))
bars = ax.bar(categories, values, color=colors, width=0.5)

# Labels on bars
for bar, val in zip(bars, values):
    ax.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 1,
            str(val), ha="center", va="bottom", fontsize=10)

# Axes & grid
ax.set_title("Evolutionary Mechanisms", fontsize=14)
ax.set_xlabel("Categories", fontsize=11)
ax.set_ylabel("Values", fontsize=11, color="blue")
ax.tick_params(axis="y", labelcolor="blue")
ax.set_ylim(0, 90)
ax.yaxis.grid(True, linestyle="--", color="lightgray")
ax.set_axisbelow(True)

plt.tight_layout()
plt.savefig("/workspace/generated_chart.png", dpi=300)
plt.show()
```

$\mathcal{C} \rightarrow \mathcal{V}$





1. Data range inaccuracy: The model-generated visualization code explicitly set data range to 0-90, which is apparently diferent from the original chart, resulting in inaccurate bar chart visualization scale.
2. Color encoding inconsistency: The visualization code fails to preserve the original color of the Y-ticks title, producing colors that also cover the Y-axis ticks.

Figure 18: **Cross-Representation Understanding Failure Example of `Claude-4.6-Sonnet`.** A case study on `Claude-4.6-Sonnet` cross-representation understanding failures. This example is a simpler bar chart, with value labels to support accurate data recognition and extraction. Nevertheless, small errors still exist even for larger-size models, such as inaccurate data range and color encoding, resulting in data visualization inaccuracies and cross-representation inconsistencies. Constraints are omitted for clarity.