Boosting Chart-to-Code Generation in MLLM via Dual Preference-Guided Refinement

Zhihan Zhang
Singapore Management University
Singapore, Singapore
zhihanzhang.2024@phdcs.smu.edu.sg

Yixin Cao
Fudan University
Shanghai, China
caoyixin2011@gmail.com

Lizi Liao
Singapore Management University
Singapore, Singapore
lzliao@smu.edu.sg

Abstract

Translating chart images into executable plotting scripts—referred to as the chart-to-code generation task—requires Multimodal Large Language Models (MLLMs) to perform fine-grained visual parsing, precise code synthesis, and robust cross-modal reasoning. However, this task is inherently under-constrained: multiple valid code implementations can produce the same visual chart, and evaluation must consider both code correctness and visual fidelity across diverse dimensions. This makes it difficult to learn accurate and generalizable mappings through standard supervised fine-tuning. To address these challenges, we propose a dual preference-guided refinement framework that combines a feedback-driven, dual-modality reward mechanism with iterative preference learning. Our approach introduces a structured variant generation strategy and a visual reward model to efficiently produce high-quality, aspect-aware preference pairs—making preference collection scalable and supervision more targeted. These preferences are used in an offline reinforcement learning setup to optimize the model toward multi-dimensional fidelity. Experimental results show that our framework significantly enhances the performance of general-purpose open-source MLLMs, enabling them to generate high-quality plotting code that rivals specialized chart-centric models and even some proprietary systems. The code and datasets are publicly available at https://github.com/Zhihan72/Chart2Code.

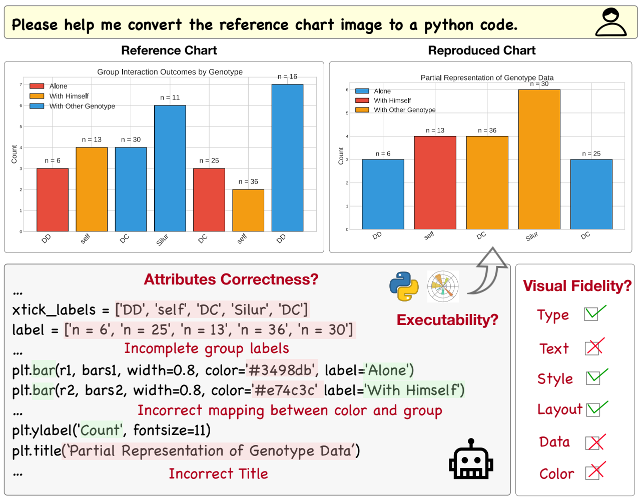
CCS Concepts

• **Computing methodologies → Natural language generation**; *Computer vision tasks*; *Reinforcement learning*.

Keywords

Multimodal Large Language Model, Chart-to-Code Generation, Offline Reinforcement Learning, Reward Modelling.

ACM Reference Format:
Zhihan Zhang, Yixin Cao, and Lizi Liao. 2025. Boosting Chart-to-Code Generation in MLLM via Dual Preference-Guided Refinement. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746027.3755596



```
...
xtick_labels = ['DD', 'self', 'DC', 'Silur', 'DC']
label = ['n = 6', 'n = 25', 'n = 13', 'n = 36', 'n = 30']
...
plt.bar(r1, bars1, width=0.8, color='#3498db', label='Alone')
plt.bar(r2, bars2, width=0.8, color='#e74c3c' label='With Himself')
...
plt.ylabel('Count', fontsize=11)
plt.title('Partial Representation of Genotype Data')
...
```

Figure 1: An illustrative example of the chart-to-code generation task, evaluated in dual-modality across multiple aspects.

1 Introduction

Charts serve as an essential medium for conveying structured information through visual representations, incorporating diverse visual elements such as colors, textual annotations, legends, and multi-panel subplots. While charts are widely used across scientific and analytical domains, understanding and reasoning over them remains a significant challenge in multimodal research [19]. Recent advancements in Multimodal Large Language Models (MLLMs) have demonstrated remarkable capabilities in addressing a wide range of chart tasks, including chart question answering [22] and chart-to-text generation [15, 16, 30]. However, these tasks typically focus on high-level semantic understanding while overlooking the intricate visual structures embedded in charts, thereby limiting their evaluation depth and applicability. In response, the chart-to-code generation task has emerged [28, 40], which requires MLLMs to jointly perform fine-grained visual parsing, accurate code synthesis, and robust cross-modal reasoning from a chart image.

The chart-to-code generation task is inherently under-constrained, presenting unique challenges for MLLMs to learn accurate and generalizable transformations from visual inputs to executable code [31]. First, a plotting code and its rendered chart do not follow a one-to-one correspondence—multiple functionally correct implementations can produce the same chart while differing in syntax, plotting logic, or visual configuration. This inherent ambiguity limits the effectiveness of standard supervised fine-tuning (SFT), which relies on exact matches to a single reference and fails to account for the diversity of valid outputs across both code and visual modalities. Second, chart construction depends on a complex combination of visual aspects—including chart type, color, text, layout, style, and underlying data—that together determine the chart's appearance and semantics. This diversity further complicates the learning objective, as models must capture subtle variations across multiple dimensions to produce faithful code. Recent studies have shown that existing open-source MLLMs often generate incorrect or non-executable code with limited alignment to the input chart [44]. These findings highlight the urgent need for an effective training paradigm that can align MLLMs with the specific demands of chart-to-code generation.

To tackle these challenges, we propose **Chart2Code**, a dual preference-guided refinement framework designed to better align the training objective with the ultimate goal of chart-to-code generation task: *generating executable code that faithfully reproduces the target chart*. The framework is built upon two key components: a *dual rewarding mechanism* and an *iterative preference learning method*. The *dual rewarding mechanism* provides fine-grained supervision by evaluating model-generated outputs across both code and image modalities. The code-side evaluation assesses the structural integrity and semantic correctness of the generated plotting script, while the image-side evaluation focuses on visual fidelity, measuring how well the rendered chart preserves the layout, styling, and perceptual attributes of the reference visualization. Leveraging this dual-modality feedback, we introduce an *iterative preference learning method* that progressively improves the model's performance. This offline Reinforcement Learning (RL) paradigm allows the evaluation of model-generated outputs against external synthetic codes through dual reward signals. The resulting preference pairs are then used to fine-tune the model via the Direct Preference Optimization (DPO) objective [27]. At the end of each iteration, the updated model is evaluated on a new batch of reference charts for the subsequent iteration, enabling continuous refinement through dual-feedback-driven optimization.

To enhance the effectiveness of preference learning, we develop a structured variant generation strategy and implement a visual reward model trained on a fine-grained, aspect-level feedback dataset during preference construction. The variant generation process produces synthetic code samples with controlled deviations from the gold-standard, enabling the creation of preference pairs that span varying levels of reproduction. To support accurate and interpretable image-side evaluation, we construct a feedback dataset comprising aspect-specific explanations and scores, which serve as supervision for training the visual reward model. This model enables reliable scoring of visual outputs, reinforcing the preference learning framework with fine-grained, aspect-aware guidance.

We validate our Chart2Code framework on three base MLLMs across two benchmarks and multiple evaluation metrics. Results demonstrate that our framework consistently yields substantial performance improvements under varying initialization settings, effectively aligning model outputs with the goal of generating high-quality, visually faithful plotting code—achieving performance comparable to specialized chart-centric models and even some proprietary systems. Our **main contributions** are:

- We propose a dual preference-guided refinement framework (Chart2Code) that aligns MLLMs to the chart-to-code task via iterative preference learning.
- We design a structured variant generation method and train a visual reward model, enabling high-quality, aspect-aware preference supervision across code and image modalities.
- We achieve performance gains across multiple MLLMs and benchmarks, showing that our method boosts general-purpose models to match or surpass chart-specific and proprietary systems.

2 Related Works

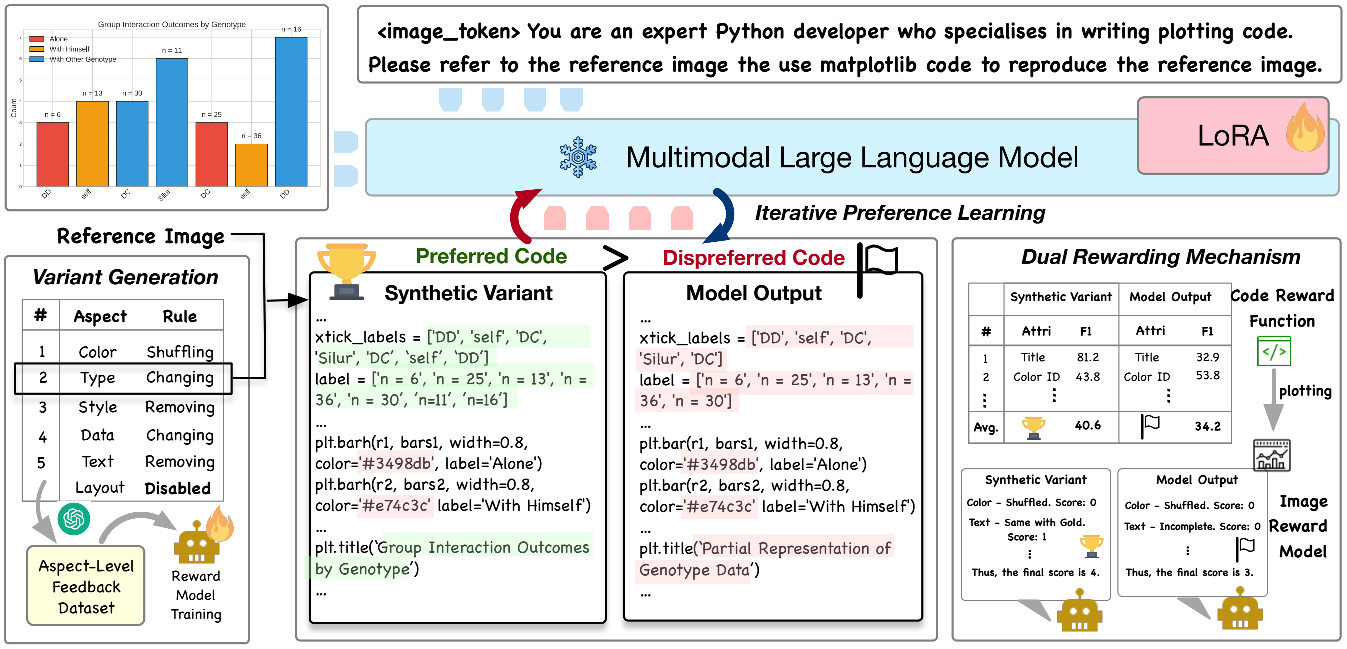
Multimodal Large Language Models. MLLMs have emerged as a transformative paradigm in artificial intelligence, enabling joint reasoning over visual and textual inputs for a wide range of cross-modal tasks [6, 41]. Building upon the success of large language models (LLMs), recent efforts have focused on aligning visual and textual representations within a shared embedding space to facilitate effective multimodal understanding [10, 39]. In parallel, there has been increasing interest in extending LLMs with multimodal instruction-following capabilities, allowing them to generate contextually grounded responses conditioned on both visual content and textual prompts [42, 47].

Preference Learning. Preference learning has emerged as a prominent approach to enhance the performance of LLMs by aligning them with human preferences, serving as the foundation of RL from Human Feedback (RLHF) [5, 25, 27]. Recent advancements of offline preference optimization techniques such as DPO [27], are becoming more popular for their simplicity and efficiency. Iterative variants of these offline methods have demonstrated effectiveness in progressively refining model outputs through repeated optimization over newly constructed preference pairs [2, 26, 38]. Although RLHF has been extensively explored in LLMs, its adaptation to MLLMs remains relatively underexplored. Existing approaches for MLLM alignment typically construct preference datasets either by leveraging externally annotated synthetic examples [17, 29, 45] or by employing self-sampling with reward-based ranking [8, 11, 43, 46]. In the chart-to-code generation setting, we combine model-generated codes with synthetic variants for iterative preference learning, exploring how this hybrid approach enhances model performance.

Chart-to-code Generation. The chart-to-code generation task has recently attracted growing attention in the research community [13, 40, 44], which requires models to synthesize executable code grounded in fine-grained visual understanding. Prior work has focused on constructing benchmarks through large-scale chart collection and human annotation to evaluate model performance in this setting [28, 35]. While several studies have explored enhancing the MLLM's chart-to-code generation capability via SFT on curated datasets [12, 20, 44], our work is the first to introduce the offline RL paradigm to align MLLMs with the inherently dual-modality and multi-dimensional requirements of chart-to-code generation.

3 Our Method: Chart2Code

We propose **Chart2Code**, a novel dual preference-guided refinement framework designed to enhance the chart-to-code generation capabilities of MLLMs by better aligning the training objective



<image_token> You are an expert Python developer who specialises in writing plotting code. Please refer to the reference image the use matplotlib code to reproduce the reference image.

#	Aspect	Rule
1	Color	Shuffling
2	Type	Changing
3	Style	Removing
4	Data	Changing
5	Text	Removing
	Layout	**Disabled**

Synthetic Variant

```
...
xtick_labels = ['DD', 'self', 'DC',
'Silur', 'DC', 'self', 'DD']
label = ['n = 6', 'n = 25', 'n = 13', 'n =
36', 'n = 30', 'n=11', 'n=16']
...
plt.barh(r1, bars1, width=0.8,
color='#3498db', label='Alone')
plt.barh(r2, bars2, width=0.8,
color='#e74c3c' label='With Himself')
...
plt.title('Group Interaction Outcomes
by Genotype')
...
```

Model Output

```
...
xtick_labels = ['DD', 'self', 'DC',
'Silur', 'DC']
label = ['n = 6', 'n = 25', 'n = 13', 'n =
36', 'n = 30']
...
plt.bar(r1, bars1, width=0.8,
color='#3498db', label='Alone')
plt.bar(r2, bars2, width=0.8,
color='#e74c3c' label='With Himself')
...
plt.title('Partial Representation of
Genotype Data')
...
```

	Synthetic Variant		Model Output	
#	Attri	F1	Attri	F1
1	Title	81.2	Title	32.9
2	Color ID	43.8	Color ID	53.8
⋮	⋮		⋮	
Avg.		40.6		34.2

Synthetic Variant
Color - Shuffled. Score: 0
Text - Same with Gold. Score: 1
⋮
Thus, the final score is 4.

Model Output
Color - Shuffled. Score: 0
Text - Incomplete. Score: 0
⋮
Thus, the final score is 3.

Figure 2: Overview of Chart2Code. It consists of two core components: a dual rewarding mechanism that provides fine-grained feedback via a heuristic F1-based code scorer and a visual reward model, and an iterative preference learning process that refines the model through DPO optimization. To enable high-quality preference construction, we introduce a structured variant generation strategy and an aspect-level feedback dataset for training the visual reward model in image-side evaluation.

with the task's inherently dual-modality and multi-dimensional nature. The framework comprises two core components: 1) *dual-modality and fine-grained rewarding mechanism*, which delivers fine-grained feedback from both image and code perspectives, guiding the model's output refinement through multi-dimensional evaluation; and 2) *iterative preference learning process*, which adopts an offline RL paradigm to collect dual-modality feedback on model outputs and progressively refine the model in subsequent iterations.

3.1 Problem Setting

Given a chart image I_i^g and an instruction x_i ($i \in [1, N]$), where g denotes the gold-standard reference and N represents the size of the gold code dataset, the target model M_t is tasked with generating code C_i^0 to replicate the reference image, where t is an integer to indicate the current iteration ($t \in [0, T]$). Formally,

$$C_i^0 = M_t(I_i^g, x_i).$$

This chart-to-code generation task is typically approached with supervised fine-tuning (SFT), where models are trained to mimic gold-standard scripts C_i^g [7, 21, 44]. However, due to the under-constrained nature of the task—where multiple valid code implementations can yield visually similar charts—SFT often fails to provide sufficiently flexible supervision needed to generalize beyond the reference outputs.

Motivated by recent advancements in RL and preference-based optimization [11, 27], we adopt a preference learning framework that allows the model to learn from relative comparisons between diverse outputs. By incorporating fine-grained, dual-modality reward signals r_i—evaluating both the generated code C_i^0 and its rendered image I_i^0—this approach aligns the training objective more closely with the true goal: *generating executable code that faithfully reproduces the target chart.*

3.2 Dual Rewarding Mechanism

3.2.1 Dual Modality. We propose a dual rewarding mechanism that provides robust and comprehensive supervision by jointly evaluating both the generated code C_i^0 and its corresponding rendered image I_i^0. This mechanism produces a code-side reward r_i^C and an image-side reward r_i^I, each reflecting different but complementary aspects of output quality. The code-side evaluation focuses on the internal structure and semantic correctness of the generated script. It verifies whether the code uses appropriate plotting APIs, encodes data relationships correctly, and adheres to syntax and logic constraints. In contrast, the image-side evaluation emphasizes external fidelity to the reference visualization, assessing how accurately the rendered chart reproduces the intended layout, styling, and perceptual attributes. This includes aspects such as spatial arrangement of subplots, font size and position of text labels, color mapping, and visual balance—features that may not be explicitly captured in the code structure but are critical to the visual appearance of the chart.

Using this dual reward mechanism, we construct high-quality preference pairs to guide the preference learning process. Specifically, we compare two generated outputs C_i^x and C_i^y ($x \neq y$) and retain the pair only if one sample strictly outperforms the other one in both reward modalities—that is, $r^C_{i,x} > r^C_{i,y}$ and $r^I_{i,x} > r^I_{i,y}$. This requirement ensures that preference labels reflect consistent superiority across both the code and image perspectives.

3.2.2 Fine-grained Reward. We introduce the rewarding methods for both the code and image modalities to enable fine-grained supervision during preference learning. On the code side, we first execute each code sample C^j_i to ensure it uses appropriate plotting APIs and adheres to basic syntactic and logical constraints. If execution fails, the sample is assigned a reward score of zero in both modalities. For executable samples, we compute the code-side reward using the heuristic F1-based scoring method [28]. This lightweight approach avoids complex code reasoning by tracing key semantic attributes during execution—such as color identifiers, titles, and data tables—which are individually compared to their counterparts in the gold-standard code using F1 score computation. The final code-side reward $r^C_{i,j}$ is calculated as the average of these attribute-level F1 scores with a range of 0–100.

On the image side, we evaluate the visual fidelity of the rendered chart I^j_i by comparing it to the reference image I^g_i using a multi-aspect binary scoring method. This method assigns a binary sub-score (0 or 1) to each of six predefined visual aspects: *chart type*, *data*, *layout*, *color*, *text*, and *style*. Each sub-score reflects whether the corresponding aspect in the generated chart aligns with the reference, considering all relevant visual cues. The total image-side reward is computed as the sum of aspect-specific sub-scores: $r^I_{i,j} = \sum_{k=1}^{6} r^I_{i,j,k}, \quad r^I_{i,j,k} \in 0, 1$. To automate this fine-grained evaluation, we train a visual reward model M_e to predict the aspect-level sub-scores based on visual differences between the generated and reference charts. The model is trained on an aspect-level feedback dataset, as described in Section 4.2.

3.3 Iterative Preference Learning

We adopt an offline RL paradigm to iteratively guide the model toward generating executable code that faithfully reproduces the target chart. In each iteration, we begin with a set of gold-standard (code, image, instruction) triplets, denoted as $\mathcal{D}^g_t = \{(C^g_i, I^g_i, x_i)\}^N_{i=1}$, along with the current model checkpoint from the previous iteration, M_t. Based on generated outputs, we construct a dataset of generated samples $\mathcal{D}^v_t = \{D^v_i\}^N_{i=1}$, where each instance D^v_i consists of a collection of code–image–reward tuples:

$$D^v_i = \{(C^j_i, I^j_i, r^C_{i,j}, r^I_{i,j}) \mid 0 \le j \le k\},$$

with each C^j_i representing either a model-generated or synthetically perturbed code sample (detailed in Section 4.1), and k indicating the total number of samples associated with the i-th reference example. Each code sample is paired with its corresponding rendered chart image and evaluated using the dual rewarding mechanism, resulting in code-side ($r^C_{i,j}$) and image-side ($r^I_{i,j}$) reward scores.

To construct the preference dataset, we pairwise all combinations of code samples in D^v_i and select the preferred sample in each pair based on their dual reward scores. This results in a preference-labeled dataset:

$$D^p_i = \{(I^g_i, x_i, C^{w_m}_i, C^{l_m}_i) \mid 1 \le m \le \tfrac{n(n-1)}{2}\},$$

where $C^{w_m}_i$ and $C^{l_m}_i$ denote the winning and losing code samples, respectively, and n is the total number of samples in D^v_i. The number of possible preference pairs is upper-bounded by $n(n-1)/2$. To ensure the quality of supervision, we discard any pair where the two samples receive identical scores in either the code-side or image-side evaluation, thereby enforcing strict agreement across both modalities.

Using the preference pairs, we train a new model M_θ , leveraging the previous iteration's model M_t as the reference model in the denominator of DPO loss function [27]. The model parameter θ is updated as follows:

$$\mathcal{L}_{DPO}(C^{w_m}_i, C^{l_m}_i | I^g_i, x_i) = \\ -\log\sigma\left(\beta\frac{M_\theta(C^{w_m}_i|I^g_i, x_i)}{M_t(C^{w_m}_i|I^g_i, x_i)} - \beta\frac{M_\theta(C^{l_m}_i|I^g_i, x_i)}{M_t(C^{l_m}_i|I^g_i, x_i)}\right),$$

where σ is the sigmoid function. At the end of this training, we obtain the updated model $M_{t+1} = M_\theta$, which is then used to generate data for the subsequent iteration.

In practice, RLHF typically begins by fine-tuning a pre-trained model on high-quality, task-specific data using supervised learning, resulting in an initial model $M_0 = M_{\text{SFT}}$ [27, 37]. Following this paradigm, we train our model on a set of gold-standard samples $\mathcal{D}^g_0 = \{(C^g_i, I^g_i, x_i)\}$ prior to initiating our Chart2Code framework, to mitigate distributional mismatch between the true reference distribution and the policy distribution used during DPO.

4 Preference Construction

To support effective preference learning, we develop a structured variant generation strategy that produces code variants with controlled levels of deviation from the gold-standard code. In parallel, we collect a feedback dataset containing detailed explanations and aspect-level annotations, which serves as supervision for training our visual reward model used in fine-grained image-side evaluation.

4.1 Rule-based Variant Generation

The rule-based variant generation strategy constructs a set of code variants that exhibit controlled levels of deviation from a given gold-standard script. Starting from a predefined set of visual aspects and their corresponding transformation rules, we sample a variation path and employ GPT-4o to generate code variants through progressive, aspect-level modifications. These synthetically perturbed variants, together with model-generated outputs, are used to construct preference pairs, effectively bridging the gap between ideal references and real model behavior.

4.1.1 Aspects and Rules. We develop a structured variant generation strategy grounded in six well-defined aspects $A = \{a_k : k \in [1, 6]\}$, capturing the full spectrum of differences that can arise between two charts. Specifically, *type* refers to the detailed chart format (e.g., donut pie chart, stacked bar chart); *data* focuses on the structure and values of the underlying dataset; *layout* captures the arrangement and number of subplots; *color* evaluates the color schemes applied to different data groups; *text* includes all textual elements such as axis labels and titles; and *style* pertains to aesthetic properties such as grid lines, borders, and marker shapes.

To enable controlled perturbations along each aspect, we define a corresponding set of transformation rules $R_k = \{r_{j,k} : j \in [1, n_k]\}$ for each a_k. These rules operate on the reference code C_i^g to generate diverse variants by selectively modifying, replacing, or removing relevant components. For example, *type* transformations are guided by a predefined dictionary mapping each chart type to its alternatives; *data* modifications involve deleting, altering, or fabricating data groups or dimensions; *layout* changes include rearranging or omitting subplots; *text* variations affect group labels, titles, and axis annotations through rewriting or removal; *color* alterations involve shuffling color schemes or reducing color diversity; and *style* adjustments toggle visual elements such as grids, borders, or legends. This structured and interpretable variation strategy allows us to generate aspect-specific variants that support consistent preference supervision and effective model training. The complete definition of aspects and rules are detailed in supplementary material.

4.1.2 Variation Path Sampling. Given the defined aspects and associated transformation rules, we sample a variation path for each gold-standard code, consisting of a sequence of aspect-rule pairs. Formally, given a reference code C_i^g, we first sample a sequence of distinct aspects $A_i = \{a_{i,\hat{k}} : 1 \le \hat{k} \le 6\}$, where each $a_{i,\hat{k}}$ is uniquely selected from the defined aspect set. For each selected aspect, we randomly sample a transformation rule $u_{i,\hat{k}} \in R_{a_{i,\hat{k}}}$, resulting in a variation path defined as $P_i^v = \{(a_{i,\hat{k}}, u_{i,\hat{k}}) : 1 \le \hat{k} \le 6\}$. Certain aspects, such as *type* and *layout*, may be inapplicable depending on the chart (e.g., non-editable types or absence of multiple subplots), resulting in variation paths ranging from 4 to 6 steps in length.

4.1.3 Variant Generation. For each gold-standard code C_i^g, we leverage GPT-4o [24] to generate structured code variants along two randomly sampled variation paths, following the self-instruct paradigm [34]. Given a variation path P_i^v, we iteratively apply each transformation rule in sequence: at step $\hat{k}$, GPT-4o is provided with the most recent variant $C_i^{\hat{k}-1}$, the selected rule $u_{i,\hat{k}}$, and an instruction prompt to generate the next variant $C_i^{\hat{k}}$. This perturbation process produces a set of progressively modified variants $\{C_i^{\hat{k}} : 1 \le \hat{k} \le 6\}$, where each variant differs from the original reference code by $\hat{k}$ aspect-level transformations. This structured generation strategy enables consistent, interpretable preference ranking across varying levels of visual fidelity. A complete example of the variation path, variant generation, and the prompt design is provided in the supplementary material.

4.2 Aspect-level Feedback Collection

To train the visual reward model for our multi-aspect binary scoring mechanism (Section 3.2.2), we construct an aspect-level feedback dataset derived from the transformation history of code variants along sampled variation paths [37]. This dataset captures detailed explanations of how each variant diverges from its reference across specific visual aspects, enabling the reward model to learn fine-grained, aspect-aware reasoning for evaluating visual fidelity in a structured and interpretable manner.

4.2.1 Feedback Composition. Each training instance in the feedback dataset is structured as: (`Reference Image`, `Task Instruction`, `Generated Image`, `Evaluation Criteria`, `Score`, `Explanation`). `Evaluation Criteria` prompts the reward model to assign a binary sub-score to each of the six predefined visual aspects of the `Generated Image`, compared with `Reference Image`. The highlighted outputs, `Score` and `Explanation`, correspond to the reward model's expected predictions: six binary sub-scores and detailed, aspect-level explanation for each decision, forming the basis of supervised training for the visual reward model.

Table 1: Distribution of numbers of gold codes, variants, and preference (Pref.) pairs across iterations.

Phrase	Gold Code	Variant	Pref. Pair
Iteration 1	300	2,752	7,802
Iteration 2	300	2,710	7,680
Iteration 3	300	2,694	7,590
Total	900	11,906	23,072

4.2.2 Feedback Collection. To construct training data for the reward model, we extract aspect-level scores and explanations from the transformation history of code variants generated along sampled variation paths by GPT-4o (Section 4.1.3). For example, consider a variant located at the third step of a variation path, where the modified aspects include layout (step 1), text (step 2), and style (step 3). We retrieve the corresponding explanations for layout and text from their respective transformation steps, and collect the explanation for style at the current step. Each of these deviated aspects is assigned a binary score of 0 to reflect a mismatch with the reference image. The rest aspects, which remain unaltered, are assigned a score of 1 and paired with a standardized explanation: The response meets the requirements in this aspect. To ensure consistency and clarity across the dataset, each feedback instance is further refined using GPT-4o to produce well-structured, aspect-specific justifications in a unified format. A complete example of feedback collection is detailed in supplementary material.

4.3 Dataset Statistics

4.3.1 Source Data. Our data source consists of the plotting scripts of ReachQA training set (3,249) [13] and ChartCoder-160k [44], each serving as a different SFT initialization setting for our Chart2Code framework. Moreover, we employ the self-instruct method [4] through GPT-4o to generate gold-standard code-image pairs $\mathcal{D}_t^g$ for each iteration. The detailed prompt is provided in the supplementary material.

4.3.2 Preference Dataset Details. Our dataset comprises 11,906 variants, and 23,072 preference pairs (including model's outputs), with their distribution across three iterations summarized in Table 1. Notably, the feedback dataset consists of 3,750 instance, with 10% reserved for evaluation. During code generation, non-executable codes are discarded, accounting for 3.9% of the total (excluded from the above counts). For each gold code, we sample two variation paths, with a maximum path length of five. The proportion of paths involving each aspect is as follows: 97.9% for *data*, 97.2% for *text*, 97.7% for *style*, 97.8% for *color*, 56.0% for *type*, and 17.3% for *layout*. The relatively lower proportions for type and layout are due to

Table 2: Performance of baselines and trained models across iterations in the Chart2Code framework on two chart-to-code datasets. The bold content represents the highest value in the category.

Models	ChartMimic				Plot2Code			
	Exec. Rate	Heuristic F1	GPT Conti.	Multi-Binary	Exec. Rate	Heuristic F1	GPT Conti.	Multi-Binary
Propriety Multimodal Large Language Models								
Gemini Pro Vision	64.2	45.0	38.1	3.47	66.3	18.7	43.9	3.36
Claude-3-Opus	86.4	56.0	45.4	3.62	**87.1**	27.4	51.9	**3.58**
GPT-4V	**91.4**	**74.3**	68.4	3.87	86.9	**31.4**	57.1	3.28
GPT-4o-mini	85.6	67.6	**70.0**	**3.95**	79.8	28.3	**58.6**	3.41
Chart-augmented Multimodal Large Language Models								
ChartInstruct-7B	1.3	0.4	1.8	0.07	2.5	0.7	1.1	0.05
ChartVLM-L-14B	12.0	3.9	3.4	0.18	15.9	2.0	2.3	0.15
ChartLlama-13B	**55.4**	**11.7**	**12.6**	**0.47**	**80.3**	**14.5**	**24.8**	**1.39**
Open-source Multimodal Large Language Models								
InternVL2.5-2B	48.8	21.9	22.6	1.31	54.5	11.0	21.5	1.41
InternVL2.5-8B	54.2	23.4	**32.1**	**1.78**	**79.5**	**17.2**	**40.5**	**2.18**
Qwen2-VL-2B	60.9	28.7	29.6	1.01	59.8	**17.2**	27.6	1.27
Qwen2-VL-7B	**62.2**	30.0	28.9	1.09	61.4	17.1	26.4	1.69
MiniCPM-Llama3-V2.5	58.2	**30.2**	24.2	1.21	58.4	16.2	22.3	1.33
LLaVA-v1.6-7B	55.6	23.6	20.2	1.09	60.6	12.8	20.1	1.13
Chart2Code (LLaVA-v1.6-7B)								
Initial 3k SFT (M_0)	56.2	24.8	23.2	1.38	57.6	11.6	21.5	1.09
Iteration 1 (M_1)	58.2	26.9	24.6	1.46	61.2	13.2	23.4	1.21
Iteration 2 (M_2)	62.2	**27.3**	25.5	**1.53**	64.0	17.8	25.6	1.35
Iteration 3 (M_3)	**63.2**	27.2	**25.8**	1.52	**66.8**	**19.4**	**32.8**	**1.45**
Initial 160k SFT (M_0)	71.8	62.3	35.6	2.83	73.5	22.2	39.5	2.71
Iteration 1 (M_1)	77.2	63.1	38.4	3.15	72.7	20.6	36.4	2.82
Iteration 2 (M_2)	79.6	65.3	41.0	3.09	80.8	24.4	42.2	3.28
Iteration 3 (M_3)	**84.6**	**69.2**	**42.1**	**3.38**	**83.3**	**24.8**	**48.5**	**3.64**

the limited number of images that support type modifications or involve multiple subplots.

5 Experiment

We validate our Chart2Code framework on open-source MLLMs across two benchmarks and multiple evaluation metrics. To evaluate the robustness of our framework under different training conditions, we experiment with two SFT initialization settings using 3k and 160k training examples, respectively. Furthermore, we conduct a series of ablation studies to evaluate the framework's generalizability across model architectures, as well as the individual contributions of the dual-modality reward mechanism and the preference construction strategy.

5.1 Experimental Settings

5.1.1 Model and Baselines. We evaluate a diverse set of MLLMs across two categories: (1) Proprietary models, including GPT-4o [24], GPT-4o-mini [23], Claude-3-Opus [3], and Gemini Pro Vision [32]. (2) Chart-augmented open-source models, such as ChartInstruct-7B [20], ChartLlama-13B [12], ChartVLM-L-14B [36], and ChartCoder-7B [44]. (3) Latest open-source models, including LLaVA-v1.6-7B (Mistral version) [18], InternVL2.5-2B, InternVL2.5-8B [9], Qwen2-VL-2B, Qwen2-VL-7B [33], and MiniCPM-Llama3-V2.5 [41].

We conduct our iterative training framework primarily on LLaVA-v1.6-7B, and further evaluate its effectiveness through ablation studies using InternVL2.5-2B and Qwen2-VL-7B. In addition, we employ Phi-3.5-Vision [1] as the backbone for training the visual reward model through SFT, leveraging its strong cross-modal reasoning capabilities and native support for multi-image input.

5.1.2 Evaluation Datasets and Metrics. We evaluate our models on two widely used chart-to-code benchmarks: ChartMimic [28] and Plot2Code [35], containing 500 and 132 examples respectively. For evaluation metrics, we adopt the two metrics from our dual rewarding mechanism—namely, the heuristic F1-based code scoring (Heuristic F1) and the multi-dimensional binary scoring (Multi-Binary) for image fidelity. Additionally, we employ GPT-4o continuous scoring (GPT Conti.), which has been commonly used in recent works [28, 40, 44]. This method prompts GPT-4o to assess the similarity between the generated and reference chart images using an open-ended, perception-driven evaluation process, assigning a continuous score ranging from 0 to 100 without relying on strict criteria. The detailed prompts for evaluation are provided in the supplementary material.

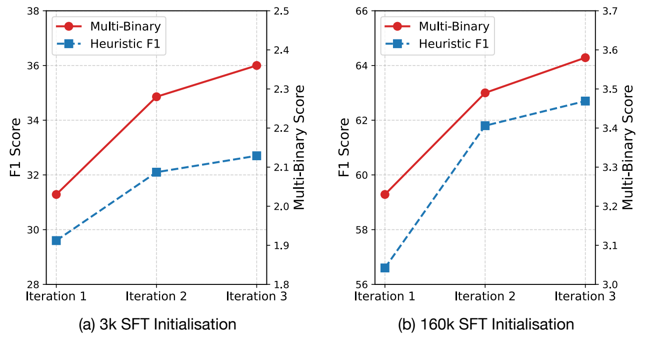


Figure 3: Rewarding signals during each iteration of Chart2Code in (1) 3k and (2) 160k SFT initialisation.

5.1.3 Implementation Details. We perform the preference learning over one epoch per iteration and evaluate two SFT initialization settings using 3k and 160k training examples, respectively, to assess the generalizability of the Chart2Code framework under varying initializing conditions (Section 4.3.1). All training runs adopt consistent LoRA fine-tuning hyperparameters [14], with `lora_r = 128` and `lora_alpha = 256`. The learning rates are set to 2e-4 for SFT and 2e-5 for DPO optimization, with a global batch size of 8 for all experiments. A complete record of training environment, settings and procesures is provided in the supplementary material.

5.2 Main Results

As shown in Table 2, our Chart2Code framework significantly and consistently improves the performance of the base MLLM across execution rate, code quality, and image fidelity under both 3k and 160k SFT initialization settings. These results demonstrate the robustness and effectiveness of our framework across varying initialization conditions. Notably, LLaVA-v1.6-7B achieves an impressive execution rate of 84.6% under the 160k initialization setting—on par with GPT-4o-mini—while also delivering substantial gains in both code quality and visual fidelity across iterations. This demonstrates that Chart2Code not only improves executability but also more effectively guides models toward generating semantically richer and structurally accurate code—surpassing the limitations of standard supervised fine-tuning.

The iteration-wise results in Table 2 demonstrate that our offline iterative preference learning strategy enables models to achieve progressively higher performance, yielding substantial improvements in both code generation and visual fidelity. This trend of refinement is also reflected in the reward signals observed across iterations. As shown in Figure 3, dual-modality reward scores exhibit consistent upward trajectories under both the 3k and 160k initialization settings, indicating steady and effective model alignment over time. Furthermore, Chart2Code delivers improvements across all evaluation dimensions, with particularly notable gains in layout, text content, and chart type accuracy. The framework also yields positive performance gains across all difficulty levels in the ChartMimic benchmark, with the most pronounced improvements observed on medium-difficulty samples. These findings are supported with further evidence in the supplementary material.

Table 3: Ablation study of base MLLMs and rewarding signal on ChartMimic.

Model	Exec. Rate	Heuri. F1	GPT Conti.	Multi-Binary
InternVL2.5-2B	48.8	21.9	22.6	1.31
Initial SFT	34.2	20.3	19.8	1.43
+ Heuristic F1	48.6	31.4	28.2	1.41
+ GPT Conti.	**56.8**	31.3	29.2	1.59
+ Multi-Binary	52.2	31.7	29.9	1.61
+ Dual Scoring	53.1	**32.7**	**31.4**	**1.66**
LLaVA-v1.6-7B	55.6	23.6	20.2	1.09
Initial SFT	56.2	24.8	23.2	1.38
+ Heuristic F1	62.2	27.0	24.8	1.41
+ GPT Conti.	62.0	25.9	24.0	1.38
+ Multi-Binary	**68.0**	26.7	25.2	1.48
+ Dual Scoring	63.2	**27.2**	**25.8**	**1.52**
Qwen2-VL-7B	62.2	30.0	28.9	1.09
Initial SFT	57.6	41.0	30.6	1.28
+ Heuristic F1	60.6	41.5	31.5	1.19
+ GPT Conti.	59.6	40.9	31.4	1.20
+ Multi-Binary	**62.8**	42.5	32.4	1.35
+ Dual Scoring	62.1	**42.9**	**33.3**	**1.36**

5.3 Ablation Study

We perform extensive ablation studies to assess the contribution of individual components within the Chart2Code framework across three base MLLMs. Owing to computational resource constraints, our ablation experiments are conducted under the 3k supervised fine-tuning initialization setting and evaluated on ChartMimic.

5.3.1 Model-agnostic Generalization. We validate the generalizability of our Chart2Code framework across three distinct MLLMs, including LLaVA-v1.6-7B, InternVL2.5-2B, and Qwen2-VL-7B, as shown in Table 3. Despite their differing architectures and capacities, all three models consistently benefit from the dual preference-guided refinement strategy, showing sustained gains in code correctness and visual alignment under all metrics. These results show that Chart2Code is not only effective but also model-agnostic—serving as a plug-and-play training framework capable of enhancing chart-to-code generation across a diverse range of MLLMs.

5.3.2 Role of Dual Rewarding. We assess the effectiveness of the dual rewarding mechanism in Chart2Code by comparing it against single-modality reward configurations. To directly evaluate the quality of the reward signals, we measure the agreement between each scoring method and the gold-standard preferences on the feedback evaluation set (Section 4.3.2), quantified by the proportion of preference pairs for which the scoring method selects the same winner as the ground truth. As shown in Table 4, the proposed dual scoring approach achieves the highest accuracy at 99.8%, followed by our multi-dimensional binary scoring method at 96.5%. While dual scoring yields a smaller set of valid preference pairs due to its stricter selection criteria, it consistently leads to stronger downstream performance across all three MLLMs on the ChartMimic benchmark (Table 3). Additionally, the proposed multi-dimensional binary scoring method outperforms GPT-4o-based scoring in both reward accuracy and downstream model performance across all evaluated MLLMs. This highlights the advantage of incorporating explicit, aspect-level reasoning in the feedback generation process,

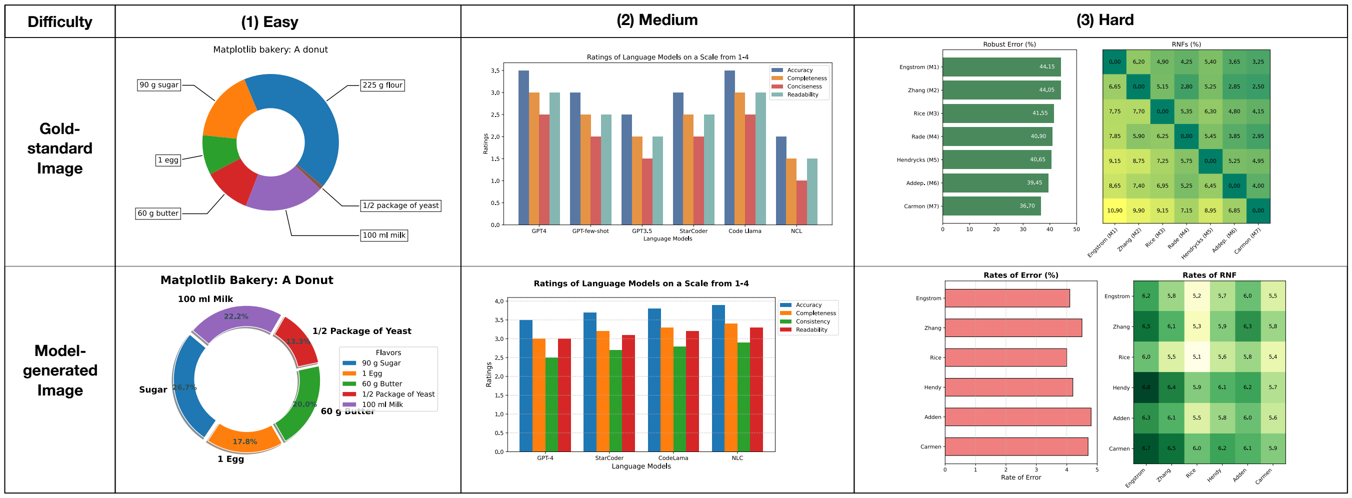


Figure 4: Case study of chart-to-code generation with Chart2Code framework. The three cases are chosen from the three difficulty levels in ChartMimic respectively.

Table 4: Rewarding accuracy (left) and drop rate (right) during preference construction under different reward signals.

Reward Signal	Prop. w. Corr. Winner	Prop. of Dropping
Heuristic F1	94.4	91.2
GPT Conti.	91.2	96.4
Multi-Binary	96.5	94.4
Dual Scoring	99.8	85.7

Table 5: Ablation study of preference learning settings.

Model	Exec. Rate	Heuri. F1	GPT Conti.	Multi-Binary
LLaVA-v1.6-7B	55.6	23.6	20.2	1.09
SFT on Gold	56.2	27.1	24.2	1.05
PL on Variants	52.4	19.3	17.6	0.60
PL on (Gold, Resp.)	49.6	24.9	22.8	0.91
Chart2Code	**63.2**	**27.2**	**25.8**	**1.52**

which produces more reliable and informative reward signals for guiding preference learning.

5.3.3 Role of Preference Construction. Our iterative training method relies on the preference construction using model-generated codes and synthetic variants, and the preference learning algorithm. To assess each component's impact, we conduct an ablation study with three settings: (1) *SFT on Gold* - supervised finetuning on all gold examples; (2) *PL on Variants* - preference learning on pairs of synthetic variants; and (3) *PL on (Gold, Resp.)* - preference learning on pairs of gold-standard codes and model-generated codes. The latter two share the same initialization as our method. As shown in Table 5, our method consistently leads to superior performance. The inclusion of model-generated codes enables the model to iteratively refine its outputs, while synthetic variants serve as a structured reference that bridges the gap between gold-standard and self-generated codes. This hybrid approach shows more effective than relying solely on gold-standard examples, fostering better adaptation and improved generalization.

6 Case Study

To qualitatively assess the MLLM's chart-to-code generation capability guided under our framework, we conduct the case study in Figure 4 showcasing three representative examples generated by LLaVA-v1.6-7B trained under the Chart2Code framework with a 3k SFT initialization. These examples are drawn from the easy, medium, and hard difficulty levels in the ChartMimic, respectively. For the easy-level donut pie chart, the model correctly identifies the chart type, color scheme, and textual elements, with only a minor stylistic deviation in legend usage. In the medium-level grouped bar chart, it preserves the overall layout and textual structure, despite omitting two data groups and mismatching some color-label associations. For the hard-level multi-panel chart, the model effectively captures the complex layout and structure, though minor inaccuracies appear in the heatmap's data and color mapping.

7 Conclusion

We presented Chart2Code, a dual preference-guided refinement framework that addresses the key challenges of chart-to-code generation—namely, the under-constrained nature of the task and the need for multi-dimensional fidelity. By combining dual-modality reward signals with structured variant generation and aspect-aware visual evaluation, our method enables scalable, fine-grained preference learning through offline reinforcement. Experiments across multiple MLLMs and benchmarks show that Chart2Code consistently improves execution accuracy and visual alignment, closing the gap between open-source and proprietary systems. Beyond these empirical gains, our work highlights the broader potential of preference-based learning in multimodal settings, especially for tasks where correctness spans both symbolic and perceptual dimensions. We believe this approach offers a generalizable path forward for aligning MLLMs with structured generation tasks.

Acknowledgments

This research is supported by the National Research Foundation, Singapore under its National Large Language Models Funding Initiative (AISG Award No: AISG-NMLP-2024-002). This research is also supported by the Ministry of Education, Singapore, under its AcRF Tier 2 Funding (Proposal ID: T2EP20123-0052). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Research Foundation or the Ministry of Education, Singapore.

References

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. arXiv:2404.14219 [cs.CL] https://arxiv.org/abs/2404.14219

[2] Leonard Adolphs, Tianyu Gao, Jing Xu, Kurt Shuster, Sainbayar Sukhbaatar, and Jason Weston. 2023. The CRINGE Loss: Learning what language not to model. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 8854–8874. doi:10.18653/v1/2023.acl-long.493

[3] Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. https://www.anthropic.com/news/claude-3-family

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[5] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomek Korbak, David Lindner, Pedro Freire, Tony Tong Wang, Samuel Marks, Charbel-Raphael Segerie, Micah Carroll, Andi Peng, Phillip J.K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. 2023. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=bx24KpJ4Eb Survey Certification, Featured Certification.

[6] Duygu Ceylan, Chun-Hao Paul Huang, and Niloy Jyoti Mitra. 2023. Pix2Video: Video Editing using Image Diffusion. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), 23149–23160. https://api.semanticscholar.org/CorpusID:257663916

[7] Jinyue Chen, Lingyu Kong, Haoran Wei, Chenglong Liu, Zheng Ge, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024. OneChart: Purify the Chart Structural Extraction via One Auxiliary Token. In *ACM Multimedia 2024*. https://openreview.net/forum?id=LagXbTuzYW

[8] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning convertsweak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 256, 22 pages.

[9] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24185–24198.

[10] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: towards general-purpose vision-language models with instruction tuning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Article 2142, 18 pages.

[11] Yihe Deng, Pan Lu, Fan Yin, Ziniu Hu, Sheng Shen, Quanquan Gu, James Zou, Kai-Wei Chang, and Wei Wang. 2024. Enhancing Large Vision Language Models with Self-Training on Image Comprehension. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=FZW7Ctyjm3

[12] Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. ChartLlama: A Multimodal LLM for Chart Understanding and Generation. arXiv:2311.16483 [cs.CV]

[13] Wei He, Zhiheng Xi, Wanxu Zhao, Xiaoran Fan, Yiwen Ding, Zifei Shan, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Distill Visual Chart Reasoning Ability from LLMs to MLLMs. arXiv:2410.18798 [cs.CL] https://arxiv.org/abs/2410.18798

[14] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=nZeVKeeFYf9

[15] Shankar Kantharaj, Xuan Long Do, Rixie Tiffany Leong, Jia Qing Tan, Enamul Hoque, and Shafiq Joty. 2022. OpenCQA: Open-ended Question Answering with Charts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 11817–11837. doi:10.18653/v1/2022.emnlp-main.811

[16] Shankar Kantharaj, Rixie Tiffany Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. 2022. Chart-to-Text: A Large-Scale Benchmark for Chart Summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 4005–4023. doi:10.18653/v1/2022.acl-long.277

[17] Lei Li, Zhihui Xie, Mukai Li, Shunian Chen, Peiyi Wang, Liang Chen, Yazheng Yang, Benyou Wang, Lingpeng Kong, and Qi Liu. 2024. VLFeedback: A Large-Scale AI Feedback Dataset for Large Vision-Language Models Alignment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 6227–6246. doi:10.18653/v1/2024.emnlp-main.358

[18] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=w0H2xGHlkw

[19] Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 2263–2279. doi:10.18653/v1/2022.findings-acl.177

[20] Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024. ChartInstruct: Instruction Tuning for Chart Comprehension and Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 10387–10409. doi:10.18653/v1/2024.findings-acl.619

[21] Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. ChartAssistant: A Universal Chart Multimodal Language Model via Chart-to-Table Pre-training and Multitask Instruction Tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 7775–7803. doi:10.18653/v1/2024.findings-acl.463

[22] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1527–1536.

[23] OpenAI. 2024. GPT-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence

[24] OpenAI. 2024. Hello GPT-4o. https://openai.com/index/hello-gpt-4o

[25] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 2011, 15 pages.

[26] Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason E Weston. 2024. Iterative Reasoning Preference Optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=4XIKfvNYvx

[27] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=HPuSIXJaa9

[28] Chufan Shi, Cheng Yang, 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. 2025. ChartMimic: Evaluating LMM's Cross-Modal Reasoning Capability via Chart-to-Code Generation. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=sGpCzsfd1K

[29] Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liangyan Gui, Yu-Xiong Wang, Yiming Yang, Kurt Keutzer, and Trevor Darrell. 2024. Aligning Large Multimodal Models with Factually Augmented RLHF. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 13088–13110. doi:10.18653/v1/2024.findings-acl.775

[30] Benny Tang, Angie Boggust, and Arvind Satyanarayan. 2023. VisText: A Benchmark for Semantically Rich Chart Captioning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 7268–7298. doi:10.18653/v1/2023.acl-long.401

[31] Leitian Tao, Xiang Chen, Tong Yu, Tung Mai, Ryan Rossi, Yixuan Li, and Saayan Mitra. 2024. CodeLutra: Boosting LLM Code Generation via Preference-Guided Refinement. arXiv:2411.05199 [cs.CL] https://arxiv.org/abs/2411.05199

[32] Gemini Team. 2024. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL] https://arxiv.org/abs/2312.11805

[33] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv preprint arXiv:2409.12191* (2024).

[34] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 13484–13508. doi:10.18653/v1/2023.acl-long.754

[35] Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. 2024. Plot2Code: A Comprehensive Benchmark for Evaluating Multi-modal Large Language Models in Code Generation from Scientific Plots. arXiv:2405.07990 [cs.CL] https://arxiv.org/abs/2405.07990

[36] Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Min Dou, Botian Shi, Junchi Yan, et al. 2024. ChartX & ChartVLM: A Versatile Benchmark and Foundation Model for Complicated Chart Reasoning. *arXiv preprint arXiv:2402.12185* (2024).

[37] Tianyi Xiong, Xiyao Wang, Dong Guo, Qinghao Ye, Haoqi Fan, Quanquan Gu, Heng Huang, and Chunyuan Li. 2024. LLaVA-Critic: Learning to Evaluate Multimodal Models. arXiv:2410.02712 [cs.CV] https://arxiv.org/abs/2410.02712

[38] Wei Xiong, Hanze Dong, Chen Ye, Han Zhong, Nan Jiang, and Tong Zhang. 2023. Gibbs Sampling from Human Feedback: A Provable KL- constrained Framework for RLHF. *ArXiv* abs/2312.11456 (2023). https://api.semanticscholar.org/CorpusID:270712238

[39] Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2022. Zero-Shot Video Question Answering via Frozen Bidirectional Language Models. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=9uRS5ysgb9

[40] Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, Zhiyuan Liu, Xiaodong Shi, and Maosong Sun. 2024. MatPlotAgent: Method and Evaluation for LLM-Based Agentic Scientific Data Visualization. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 11789–11804. doi:10.18653/v1/2024.findings-acl.701

[41] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. MiniCPM-V: A GPT-4V Level MLLM on Your Phone. arXiv:2408.01800 [cs.CV] https://arxiv.org/abs/2408.01800

[42] Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yi Zhou, Junyan Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, Chenliang Li, Yuanhong Xu, Hehong Chen, Junfeng Tian, Qiang Qi, Ji Zhang, and Feiyan Huang. 2023. mPLUG-Owl: Modularization Empowers Large Language Models with Multimodality. *ArXiv* abs/2304.14178 (2023). https://api.semanticscholar.org/CorpusID:258352455

[43] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 2389, 19 pages.

[44] Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Wanxiang Che, Zhiyuan Liu, and Maosong Sun. 2025. ChartCoder: Advancing Multimodal Large Language Model for Chart-to-Code Generation. arXiv:2501.06598 [cs.AI] https://arxiv.org/abs/2501.06598

[45] Yiyang Zhou, Chenhang Cui, Rafael Rafailov, Chelsea Finn, and Huaxiu Yao. 2024. Aligning Modalities in Vision Large Language Models via Preference Fine-tuning. arXiv:2402.11411 [cs.LG]

[46] Yiyang Zhou, Zhiyuan Fan, Dongjie Cheng, Sihan Yang, Zhaorun Chen, Chenhang Cui, Xiyao Wang, Yun Li, Linjun Zhang, and Huaxiu Yao. 2024. Calibrated Self-Rewarding Vision Language Models. *arXiv preprint arXiv:2405.14622* (2024).

[47] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2024. MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=1tZbq88f27