A Large-Scale Diverse and Complex Dataset for Enhancing Chart-to-Code Generation

Anonymous ACL submission

Abstract

Chart2Code has recently received significant attention in the multimodal community due to its potential to reduce the burden of visualization and promote a more detailed understanding of charts. However, existing Chart2Code-related training datasets suffer from at least one of the following issues: (1) limited scale, (2) limited type coverage, and (3) inadequate complexity. To address these challenges, we seek more diverse sources that better align with real-world user distributions and construct a data synthesis pipeline and further created a large-scale Chart2Code training dataset. Experimental results demonstrate that even with fewer parameters, the model finetuned on our dataset achieves state-of-the-art performance on multiple Chart2Code benchmarks within open-source models.

1 Introduction

With the development of multimodal large language models (MLLMs) (Liu et al., 2023; Wang et al., 2024; Chen et al., 2024), an increasing amount of research has applied them to Chart-related tasks (Meng et al., 2024; Zhang et al., 2024a; Han et al., 2023; Huang et al., 2024) . Chart2Code is one of them which requires the MLLM to receive a chart as input and generating source code that accurately replicates the chart. The task requires the MLLM not only to perceive the content of the chart precisely but also to organize the perceived information with appropriate code logic (Wu et al., 2024; Shi et al., 2025).

Chart2Code has recently gained significant attention because of its potential to assist in data visualization (Shi et al., 2025) and promote a more detailed understanding of charts (Xu et al., 2025). Several benchmarks have been introduced to evaluate Chart2Code (Wu et al., 2024; Shi et al., 2025). According to the evaluation results, existing open-source MLLMs still perform poorly in Chart2Code and exhibit a significant gap when compared with the closed-source models.

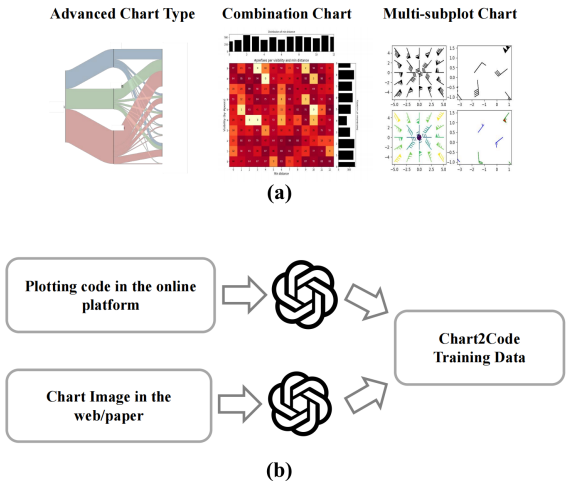

Figure 1: Our work focuses on Chart2Code task. (a) Different from existing work, we focus on creating more advanced and complex charts. (b) High-level illustration of our dataset construction pipeline. We use ChatGPT to rewrite the existing diverse web plotting code into executable code or directly instruct it to synthesize executable code based on existing chart images. The charts are obtained by executing the result code.

Currently, all the open-source Chart2Code-related training dataset[1] have at least one following issues: **(1) Limited scale:** The training samples are not enough for the model to learn the challenge task (He et al., 2024) . **(2) Limited Type Coverage:** The most diverse dataset includes only 31 chart types (Pesaran Zadeh et al., 2024), while matplotlib can generate many more types. **(3) Gap Exists with real-world user needs:** Text2Vis (Nguyen et al., 2024) points out that the existing datasets do not adequately align with the real-world requirements of the users. For example, current datasets mostly pay attention to single-type charts while ignoring complex and composite

[1]We use Chart2Code-related as there are Text2Chart training dataset which closely related to Chart2Code.

	Data form	# Chart types	# Data samples	# Matplotlib API types	# Different API combinations	# Avg code length
ChartLlama	Chart2Code	10	11K	83	418	17
ChartMOE	Chart2Code	<20	800K	-	-	-
ReachQA	Chart2Code	15	3K	168	2222	22
Text2Chart31	Text2Chart	31	11.1K	188	1881	12
Ours	Chart2Code	53	120K	1219	84214	23

Table 1: Stastics of various Chart2Code-related datasets. ChartLlama (Han et al., 2023), ChartMOE (Xu et al., 2025) and Text2Chart31 (Pesaran Zadeh et al., 2024) have relatively more training samples but limited complexity and diversity. ReachQA (He et al., 2024) has enough diveristy and complexity while having limited scale. Our dataset combines all the

charts.

To address the aforementioned issues, we aim to construct a standard Chart2Code training dataset. **To solve issue(2)**, we construct a comprehensive chart type taxonomy and synthesize data that include each type respectively. **To solve issue(3)**, we seek the source that may better reflect the user needs and propose two synthesis pipelines: **Synthesize based on online plotting code (Kocetkov et al., 2022)**, which predefines certain rules to filter relevant code snippts in Web Code and instructs GPT4 (OpenAI et al., 2024) to synthesize executable code based on them and **Synthesize based on web chart images (Li et al., 2024)**, which directly feed the selected chart images to GPT4 to synthesize the code.

We conduct analysis and compare our constructed dataset with other Chart2Code-related datasets. Our results demonstrate that our dataset encompasses **a wider variety of chart types and a more diverse distribution of complexity.** We then fine-tune an open-source MLLM (Chen et al., 2024) using our constructed data. Experimental results demonstrate that even with relatively small parameters (4B), the model fine-tuned on our data exhibits significant improvements across various Chart2Code benchmarks, achieving state-of-the-art performance compared to other open-source models.

2 Dataset construction

2.1 Task definition

Given an input chart image I and plotting instruction $\mathcal{T}$, a MLLM is required to output an executable code C.

$$C = \arg\max_{C} P_{MLLM}(C|\mathcal{T}, I) \quad (1)$$

By utilizing an external interpreter (e.g., Python), the plotting code is executed to generate an image I'.

$$I' = Interpreter(C) \quad (2)$$

The goal is to ensure I' and I as close as possible. In this work, we focus on matplotlib based charts, leaving other types for future work.

2.2 Dataset construction pipelines

2.2.1 Creating chart type taxonomy

To address the limited type coverage issue, we first construct a comprehensive chart type taxonomy by first merging the chart types specified in recent works(Xu et al., 2024; He et al., 2024; Hu et al., 2024) and then adding additional chart types given by GPT4, which result in 53 chart types. We synthesize code that **include** each type respectively.

2.2.2 Synthesize based on online plotting code

To synthesize dataset that more align with the real need of users, we first extract plotting code snippets from the Stack dataset following Text2vis(Nguyen et al., 2024). However the extracted snippts have the following issues: (1) Contain many lines unrelated to plotting. (2) The plotting logic tends to be homogeneous. (3) Most code snippets cannot be directly executed to produce chart image. To address the issues, we divide the synthesis process into three steps: **extracting, filtering, and rewriting.** Each step is designed to resolve issues (1), (2), and (3), respectively.

In the **extracting** step, for each python code file, we extract matplotlib function calls and assignment statements following text2vis. We retain relevant functions and control statements, and partition the results based on the call chain.

In the **filtering** step, for each chart type, we filter relevant code snippets based on predefined rules and use LSH within different API length ranges respectively to select code snippts, ensuring coverage of diverse plotting logics.

In the **rewriting** step, we pass the results in filtering to GPT4 to generate complete and executable plotting code, with the prompt instructing

it to faithfully replicate the user's plotting logic as accurately as possible.

2.2.3 Synthesize based on online chart images

To increase the data volume of sparse chart types and enhance the diversity of other categories, inspired by GPT4's great performance in Chart2Code, we propose to directly synthesizing code based on chart image for each chart category. Specifically, we choose Multi-modal arxiv dataset(Li et al., 2024) as our image base. We first use GPT4 to generate visual feature descriptions for each chart type. Then we filter the corresponding charts using Siglip (Zhai et al., 2023). Finally, we prompt GPT to generate the plotting code based on the images.

2.2.4 Quality control

We aim to check and control the quality of our data both in image aesthetics and code quality.

For image aesthetics, we follow the multi-modal self-instruct (Zhang et al., 2024b), using LLaVA v1.5 (Liu et al., 2023) to check for conflicts in visual elements and the rationality of the layout. We remove the image which fail to pass the checking.

For code quality, we mainly check whether the code contains anything that is unrelated to plot the chart. We randomly selected 5 samples from each chart type and found that fewer than 5% exhibit such issues. Given resource limitations, we don't deal with them.

2.3 Dataset analysis

We give detailed analysis of our dataset in this section. We show the chart type distribution of our constructed dataset in 4 and show qualitive synthsised data in appendix.

Chart type and combination diversity As shown in Table 1, our dataset is the largest among existing Chart2Code-related datasets, with 53 chart types, the most diverse of any related dataset. Additionally, due to we take more plotting resource into consideration, our dataset includes 1219 Matplotlib API types and 84,214 API combinations, both exceeding the numbers in existing datasets. **To summarize, our dataset exhibits much higher chart type and combination diveristy.**

Complexity diversity As shown in the Fig 2, the distributions of the number of Matplotlib APIs and total code length per plotting code

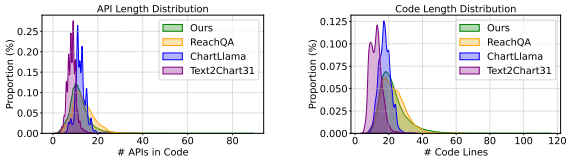

Figure 2: Matplotlib api length distribution and code length distribution.

in the ChartLlama and Text2Chart31 datasets are densely concentrated around specific points, whereas our dataset and ReachQA exhibit a more uniform distribution (although the ReachQA dataset is smaller in scale). **This demonstrates that our dataset offers a well-balanced diversity in complexity.**

3 Experiments

3.1 Experimental setup

We finetune the InternVL2-4B (Chen et al., 2024) model using our constructed dataset and evaluate the Chart2Code task using ChartMimic (Shi et al., 2025) and Plot2Code (Wu et al., 2024) benchmarks. We fully follow their evaluation pipelines.

3.2 Main results

Table 2 shows the evaluation results. We have the following conclusions.

Chart-specific models fail on the benchmark although finetuned on their own Chart2Code data. ChartMOE and TinyChart were trained on a larger-scale Chart2Code dataset and demonstrated their superiority in performing this task. However, when evaluated on these two real-world Chart2Code benchmarks, their performance showed a significant decline. This drop in performance can primarily be attributed to the insufficient diversity and complexity of the charts in the datasets they were trained on. The dataset we propose can effectively fill the gap.

Model Finetuned on our dataset achieve SOTA performance. As shown in Table 2, the InternVL2-4B model, after fine-tuning on our dataset, achieves a significant performance improvement. Moreover, it outperform other open-source model of much larger parameters and achieves SOTA performance. This strongly validates the effectiveness of our dataset. On the high-level metric of ChartMimic, the model performs slightly worse than InternVL2-Llama3-76B, despite having a significantly lower code execution success rate. We

Model Name	Params	ChartMimic				Plot2Code	
		Execute Rate	Low-Level	High-Level	Overall	Execute Rate	GPT4v rating
GeminiProVision	-	68.2	53.8	53.3	53.55	68.2	3.69
Claude-3-opus	-	83.3	60.5	60.1	60.3	84.1	3.8
GPT-4o	-	**93.2**	**79**	**83.5**	**81.25**	**88.6**	**5.71**
Qwen2-VL-7B	8.2B	47	32.9	35	33.95	68.2	3.12
InternVL2-4B	4.2B	50.5	33.8	38.4	36.1	66.3	2.52
InternVL2-8B	8.1B	52.5	34.4	38.9	36.65	77.3	2.78
MiniCPM-Llama3-V-2.5	8.4B	67	36.6	42.1	39.35	76.3	2.61
InternVL2-26B	26.0B	69.3	41.4	47.4	44.4	81.3	3.42
InternVL2-Llama3-76B	76.0B	83.2	54.8	**62.2**	58.5	83.2	54.1
TinyChart	3B	42.5	26.3	25.9	26.1	43.2	2.19
ChartMOE	7B	52.7	25.3	22.9	24.1	65.2	2.22
InternVL2-4B-Finetune(Ours)	4.2B	**78.3**	**63.4**	60.4	**61.9**	**84.8**	**4.49**

Table 2: Chart2Code results for various closed-source and open-source models. The highest scores in each model category are marked in bold. Despite having only 4B parameters, the model fine-tuned on our dataset achieves state-of-the-art performance across the evaluated benchmarks.

Model	Text	Layout	Type	Color	Avg
GPT-4o	**81.5**	**89.8**	**77.3**	**67.2**	**79**
InternVL2-26B	39.2	58.7	35.9	31.8	41.4
InternVL2-Llama3-76B	54.1	74.5	49.2	41.5	54.8
ChartMOE	24.4	42.05	18.61	16.1	25.3
InternVL2-4B-Finetune(Ours)	**61.6**	**74.9**	**62.9**	**54**	**63.4**

Table 3: Model performance across different dimensions in ChartMimic.

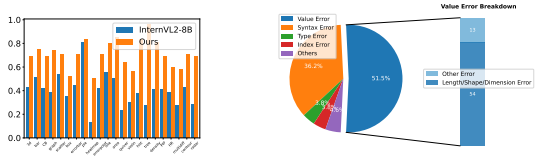

Figure 3: Matplotlib api length distribution and code length distribution.

believe that this performance gap is more likely due to the inherent limitations in code generation capabilities of the 4B parameter base model itself.

3.3 Analysis

We use our finetuned model to conduct in-depth analysis based on ChartMimic in this section.

The model shows consistent significant performance improvements across different categories. As shown in the figure 3 left, our model demonstrates significant performance gains across all chart types, including complex types such as CB and HR which are not explicitly specified in our chart taxonomy. This suggests that our dataset is well-balanced, enabling the model to better adapt to diverse and complex real-world scenarios.

The models ability to capture chart details and handle complex logic needs improvement. As shown in Table 3, our model shows a notable gap in text performance compared to GPT-4o. Additionally, all models score much lower on the color metric, indicating weaker capture of low-level details. We also find that samples with for-loops perform nearly 10% worse, suggesting the model struggles with complex plotting logic.

Most coding error of the model are Syntax errors and variable planning errors. As shown in the figure 3 right, coding errors are primarily syntax and value errors, with the latter mainly due to dimension mismatches of the variables defined before the they are used. This indicates that apart from general coding abilities, variable planning is an important ability for Chart2Code task that might be considered to be further improved, which may be challenging due to the auto-regressive nature of current MLLMs.

4 Conclusion

This paper addresses the limitations of existing Chart2Code-related datasets, including insufficient quantity, diversity, and complexity. We propose a data synthesis pipeline to create a large-scale Chart2Code training dataset and conduct fine-tuning experiments on an open-source model. The results show that the model achieves SOTA performance with fewer parameters. However, the analysis reveals that even after large-scale fine-tuning, the model's ability to perceive chart details and generate code remains a limiting factor. We hope our dataset will inspire further research in this area.

Limitations

The primary limitation of this study lies in the training dataset, which is currently restricted to the matplotlib library. While this covers a wide range

of common visualizations, it restricts the diversity of charts that can be generated, as other libraries such as seaborn, plotly, or ggplot are not included. Future work could expand the dataset to include these libraries, allowing for a broader variety of visualization code generation.

References

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, Ji Ma, Jiaqi Wang, Xiao wen Dong, Hang Yan, Hewei Guo, Conghui He, Zhenjiang Jin, Chaochao Xu, Bin Wang, Xingjian Wei, Wei Li, Wenjian Zhang, Bo Zhang, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, and Yu Qiao. 2024. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *ArXiv*, abs/2404.16821.

Yucheng Han, China. Xiaoyan 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*, abs/2311.16483.

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. *Preprint*, arXiv:2410.18798.

Linmei Hu, Duokang Wang, Yiming Pan, Jifan Yu, Yingxia Shao, Chong Feng, and Liqiang Nie. 2024. Novachart: A large-scale dataset towards chart understanding and generation of multimodal large language models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 39173925, New York, NY, USA. Association for Computing Machinery.

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

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code. *Preprint*, arXiv:2211.15533.

Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. 2024. Multimodal ArXiv: A dataset for improving scientific comprehension of large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14369–14387, Bangkok, Thailand. Association for Computational Linguistics.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26286–26296.

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*, pages 7775–7803, Bangkok, Thailand. Association for Computational Linguistics.

Hy Nguyen, Xuefei He, Andrew Reeson, Cecile Paris, Josiah Poon, and Jonathan K. Kummerfeld. 2024. Do text-to-vis benchmarks test real use of visualisations? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7433–7441, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, ukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, ukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim,

Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Fatemeh Pesaran Zadeh, Juyeon Kim, Jin-Hwa Kim, and Gunhee Kim. 2024. Text2Chart31: Instruction tuning for chart generation with automatic feedback. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11459–11480, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang 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*, abs/2409.12191.

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*, abs/2405.07990.

Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. 2024. Chartbench: A benchmark for complex visual reasoning in charts. *Preprint*, arXiv:2312.15915.

Zhengzhuo Xu, Bowen Qu, Yiyan Qi, SiNan Du, Chengjin Xu, Chun Yuan, and Jian Guo. 2025. Chartmoe: Mixture of diversely aligned expert connector for chart understanding. In *The Thirteenth International Conference on Learning Representations*.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11941–11952.

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. 2024a. TinyChart: Efficient chart understanding with program-of-thoughts learning and visual token merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1898, Miami, Florida, USA. Association for Computational Linguistics.

Wenqi Zhang, Zhenglin Cheng, Yuanyu He, Mengna Wang, Yongliang Shen, Zeqi Tan, Guiyang Hou, Mingqian He, Yanna Ma, Weiming Lu, and Yueting Zhuang. 2024b. Multimodal self-instruct: Synthetic abstract image and visual reasoning instruction using language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19228–19252, Miami, Florida, USA. Association for Computational Linguistics.

A Appendix

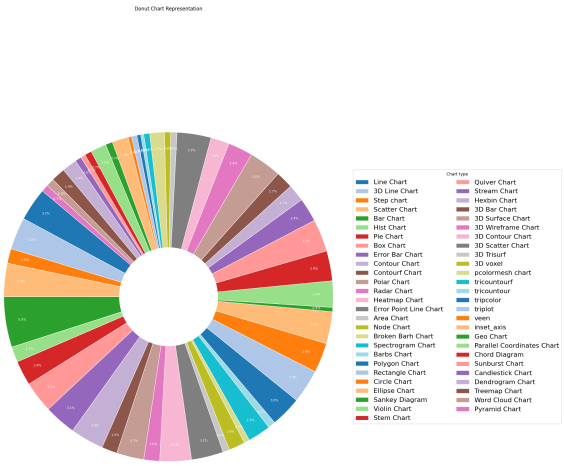

Figure 4: category distribution

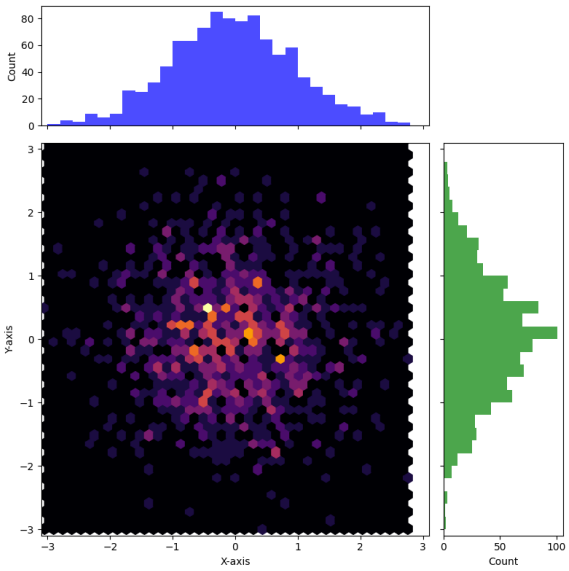

Figure 7: synthesised chart

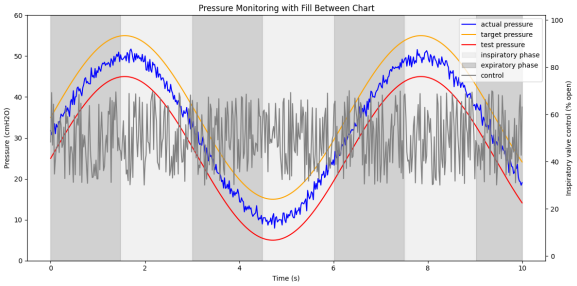

Figure 5: synthesised chart

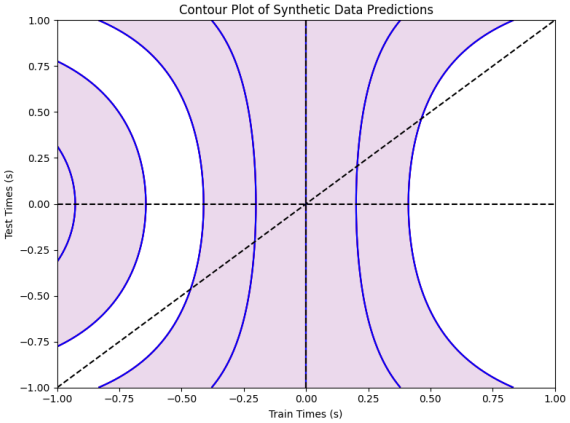

Figure 6: synthesised chart

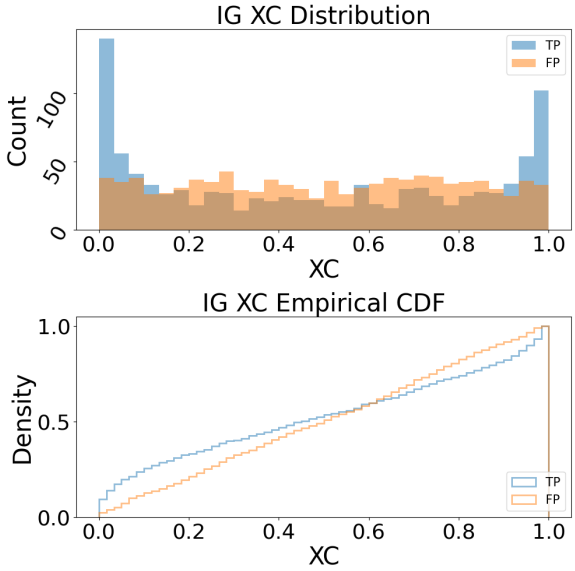

Figure 8: synthesised chart

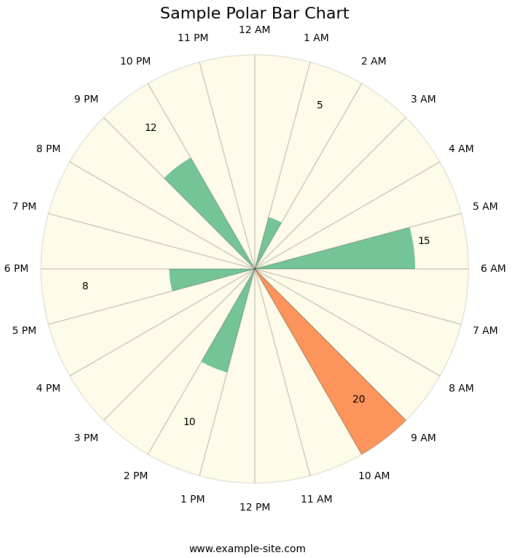

Figure 9: synthesised chart

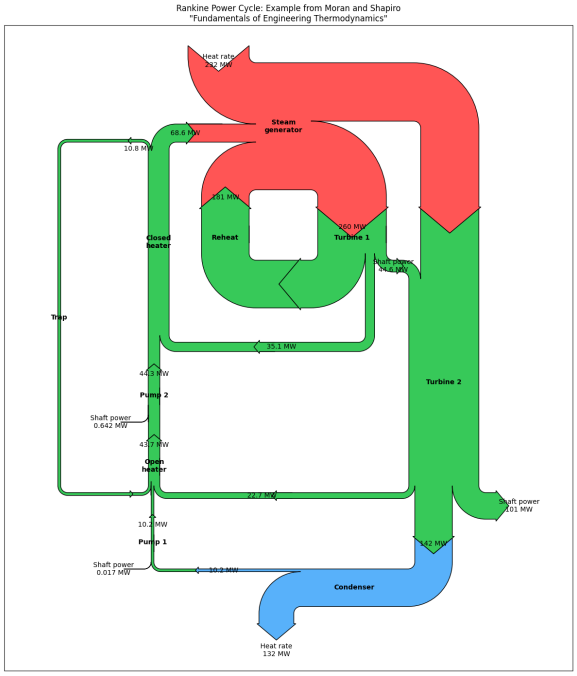

Figure 11: synthesised chart

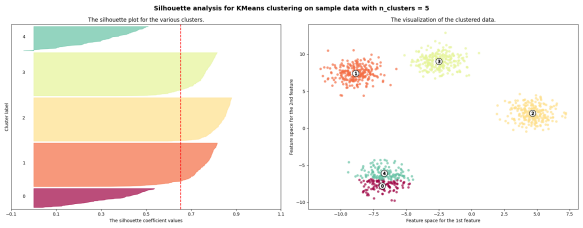

Figure 12: synthesised chart

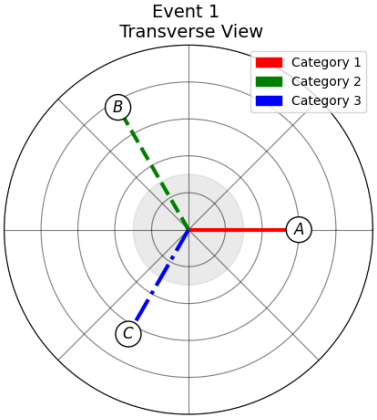

Figure 10: synthesised chart

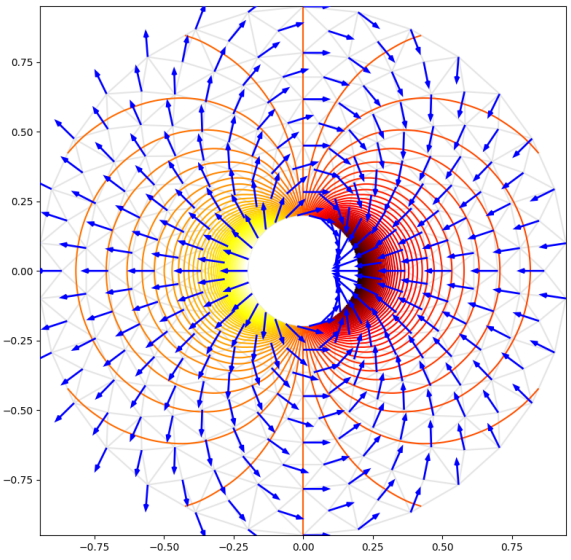

Figure 13: synthesised chart

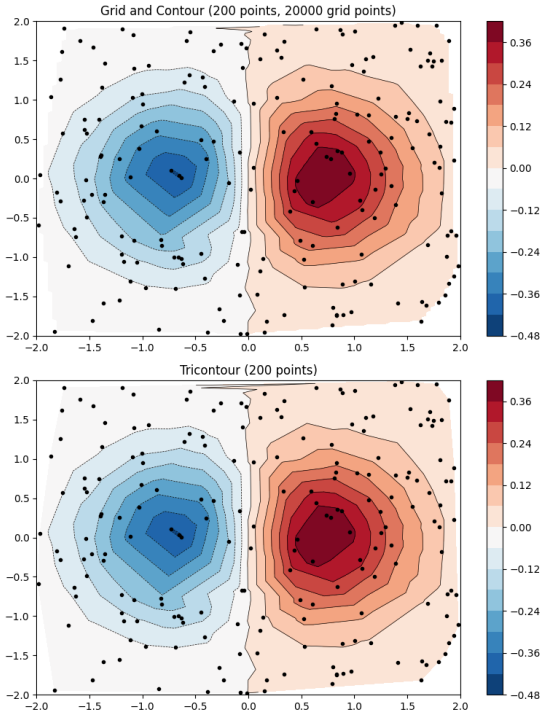

Figure 14: synthesised chart