Chart2Code53: A Large-Scale Diverse and Complex Dataset for Enhancing Chart-to-Code Generation

Tianhao Niu[♠*] **Yiming Cui**[♣] **Baoxin Wang**[♣] **Xiao Xu**[♠]
Xin Yao[♣] **Qingfu Zhu**[♠†] **Dayong Wu**[♣] **Shijin Wang**[♣] **Wanxiang Che**[♠†]
[♠] Research Center for Social Computing and Interactive Robotics
[♠] Harbin Institute of Technology, China
[♣] State Key Laboratory of Cognitive Intelligence, iFLYTEK, Beijing, China
{thniu,qfzhu,car}@ir.hit.edu.cn
ymcui@iflytek.com

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 propose dual data synthesis pipelines: (1) Synthesize based on online plotting code. (2) Synthesize based on the chart images in the academic paper. We create a large-scale Chart2Code training dataset Chart2Code53, including 53 chart types, 130K Chart-code pairs based on the pipeline. Experimental results demonstrate that even with few parameters, the model finetuned on Chart2Code53 achieves state-of-the-art performance on multiple Chart2Code benchmarks within open-source models[1].

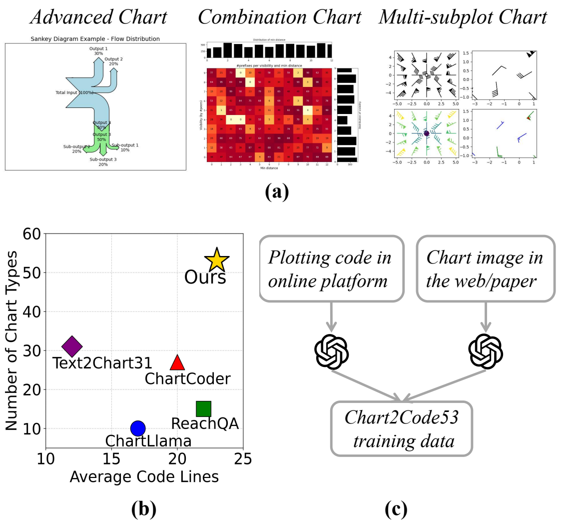

Figure 1: Our work focuses on Chart2Code task. (a) Different from existing work, we focus on creating more advanced and complex charts. (b) Compared to other existing open-source Chart2Code-related datasets, our dataset exhibits the greatest diversity and higher complexity. (c) High-level illustration of our dataset construction pipeline. We use GPT-4o 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. These two data synthesis pipelines can generate more complex and diverse Chart2Code data.

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 generate 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., 2025; 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., 2025; 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.

Currently, all the open-source Chart2Code-related training datasets have at least one following issues: **(1) Limited scale:** The training sam-

[*] The work is done during internship at iFLYTEK.
[†] Corresponding authors.
[1] Code and data: https://github.com/nth2000/Chart2Code53

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 15840–15856
November 4-9, 2025

Dataset name	Data form	# Chart types	# Data samples	# API types	# API combinations	# Avg code length
ChartLlama	Chart2Code	10	11K	83	418	17
ChartMOE	Chart2Code	<20	800K	-	-	-
ReachQA	Chart2Code	15	3K	168	2,222	22
Text2Chart31	Text2Chart	31	11K	188	1,881	12
ChartCoder	Chart2Code	27	115K	187	4,421	20
CoSyn	Chart2Code	-	53K	344	27,881	22
Chart2Code53	Chart2Code	53	130K	1,219	84,214	23

Table 1: Statistics of various Chart2Code-related datasets. While ChartLlama (Han et al., 2023), ChartMOE (Xu et al., 2025), and Text2Chart31 (Pesaran Zadeh et al., 2024) offer a relatively larger number of training samples, they exhibit limited complexity and diversity. In contrast, ReachQA (He et al., 2024) provides greater diversity and complexity but is limited in scale. Our dataset Chart2Code53 effectively integrates all these advantages. Concurrent works ChartCoder (Zhao et al., 2025) and CoSyn (Yang et al., 2025) are also listed.

ples 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.

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 includes 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 snippets in web code and instruct GPT-4o (OpenAI et al., 2024) to synthesize executable code based on them and **synthesize based on web chart images (Li et al., 2024b)**, which directly feed the selected chart images to GPT-4o to synthesize the code.

We conduct analysis and compare our constructed dataset Chart2Code53 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 (7B), 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.

The contributions of our work are summarized as follows:

- **Dual Data Synthesis Pipelines:** We propose a dual-pipeline framework for synthesizing chart-code pairs, enabling the generation of high-quality, diverse, and structurally complex training data to facilitate Chart2Code model learning.
- **Chart2Code53 Dataset:** Based on the pipeline, we construct Chart2Code53, which comprises 130K high-quality chart-code pairs spanning 53 distinct chart types, significantly surpassing previous datasets in scale, diversity, and complexity.
- **Specialized MLLM for Chart2Code:** We present an open-source MLLM tailored for Chart2Code task. Despite its compact size (7B parameters), the model outperforms all existing open-source MLLMs on Chart2Code benchmarks,

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.

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

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

$$I' = \text{Interpreter}(C) \tag{2}$$

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.

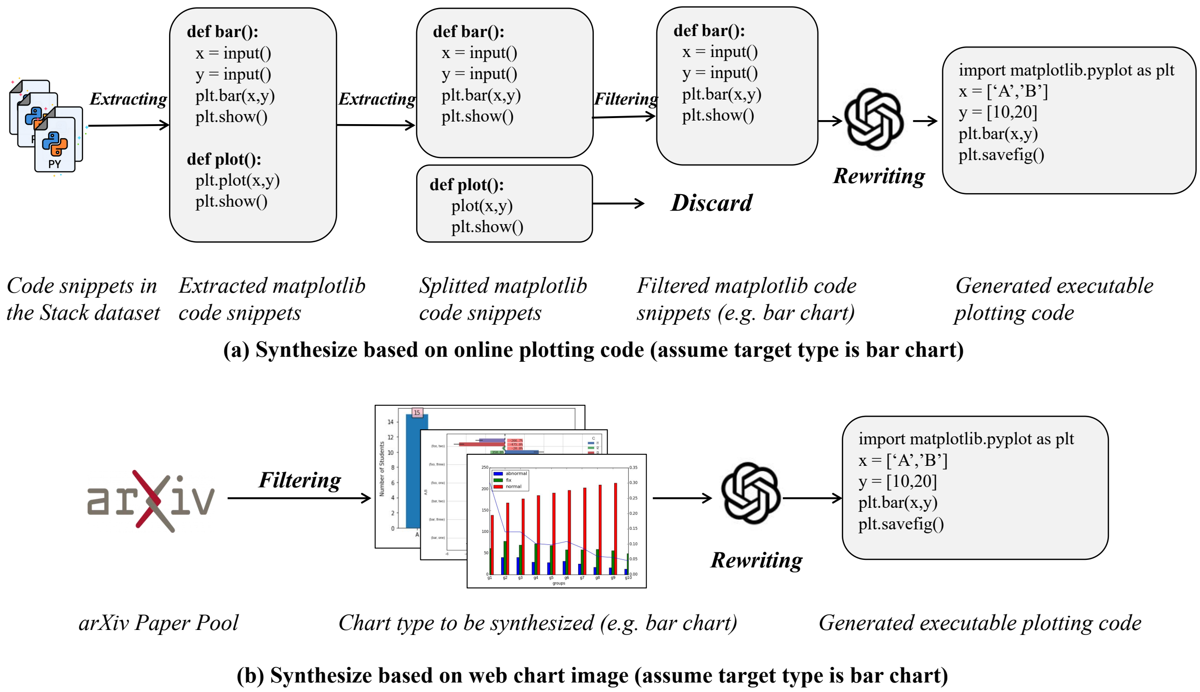

Figure 2: Overview of the dual data synthesis pipeline. (a) Synthesis based on online plotting code. (b) Synthesis based on web chart images. The generated code from each pipeline is executed and further refined through quality control.

2.2 Dataset construction pipelines

2.2.1 Overview

Our goal is to construct a large-scale, diverse, and complexity-varied training dataset for Chart2Code. To achieve this, we first establish a comprehensive chart type taxonomy. Then we employ the dual pipeline to synthesize plotting code for each type **respectively**. An overview of the dual pipeline is illustrated in Figure 2. The final plotting code is then executed by a Python interpreter to obtain the corresponding chart images. Finally, we filter the dataset by evaluating both the visual aesthetics of the images and the quality of the code. The resulting dataset is named Chart2Code53.

2.2.2 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 GPT-4o, which results in 53 chart types. We synthesize code that **includes** each type respectively.

2.2.3 Synthesize based on online plotting code

To synthesize a dataset that better aligns with real user needs, we first extract plotting code snippets from the Stack dataset following Text2vis (Nguyen et al., 2024). However, the extracted snippets 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, we first filter the plotting code snippets to retain only those matching the target chart type, using rules uniquely determined by API call patterns and parameter characteristics (e.g., selecting fragments containing .bar() function calls to match bar charts). All rules were manually verified to ensure accuracy. Subsequently, we employ a combined approach of

Locality-Sensitive Hashing (LSH) and a bucketing strategy to further refine the selection, prioritizing code snippets that exhibit both diversity and complexity.

Specifically, we distribute the code into 5 buckets with uniformly increasing length ranges based on API call sequences. Within each bucket, we apply LSH to cluster code fragments and select representatives with maximally diverse API combinations. This process ensures that the final synthesized code snippets exhibit both diversity and complexity within each chart type.

In the **rewriting** step, we pass the results of the filtering step to GPT-4o to generate complete and executable plotting code, with the prompt instructing it to faithfully replicate the user's plotting logic, including function calls, parameters, and control flows as accurately as possible. Additionally, the target chart type is specified in the prompt to prevent potential mismatches between the code snippets provided in the previous filtering step and the intended target chart type. [2]

2.2.4 Synthesize based on online chart images

To increase the data volume of sparse chart types and enhance the diversity of other categories, inspired by GPT-4o's great performance in Chart2Code, we propose to directly synthesize code based on chart image for the target chart type. Specifically, we choose Multi-modal arXiv dataset (Li et al., 2024b) as our image base. To filter the target chart type, we follow Menon and Vondrick (2023) and use GPT-4o to generate 3 distinct visual feature descriptions. Then we filter the corresponding charts using SigLIP (Zhai et al., 2023) based on the description. Then, we prompt GPT-4o to generate the plotting code based on the images. The prompt should specify the target chart type to prevent a few type mismatches between the selected chart and the expected target chart type.

2.2.5 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.

[2]This situation may occur due to unexpected boundary cases where the code snippets filtered in the previous step do not perfectly match the target chart type. Based on our sampling of 100 code snippets per chart type, we found the mismatch rate to be less than 1%.

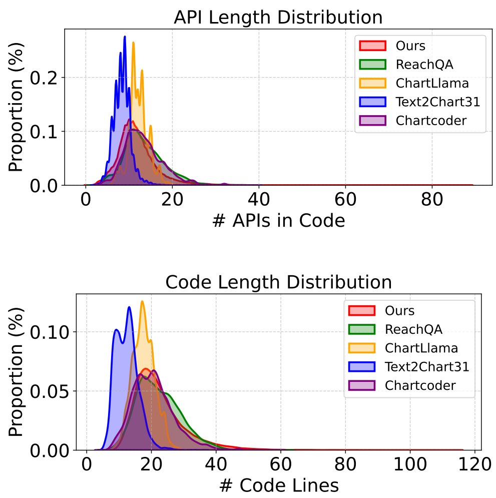

Figure 3: Matplotlib API length distribution and code length distribution.

For code quality, we mainly check whether the code contains anything that is unrelated to plotting the chart. We manually check 50 samples per category and recognize code-related issues. Only 3.2% of the data may have such problems. Given resource restrictions, we don't deal with them. Further details are shown in the Appendix.A.2.

2.3 Dataset analysis

We give a detailed analysis of Chart2Code53 in this section. We show qualitatively synthesized data in the Appendix.A.3.[3]

Chart type distribution As shown in Table 1, Chart2Code53 is the largest among existing Chart2Code-related datasets, with more chart types and API types, which is the most diverse of any related dataset. We show the chart type distribution in Figure 9. As illustrated, the distribution of categories is well-balanced.

Code complexity diversity As shown in the Fig 3, the distributions of the number of Matplotlib APIs and total code length per plotting code in the ChartLlama and Text2Chart31 datasets are densely concentrated around specific points,

[3]Although our current data doesn't include charts from other plotting packages beyond matplotlib, our pipeline can be readily adapted to other API-based visualization libraries (e.g., Plotly, ggplot) by simply incorporating their respective function names and keywords during the extraction phase - all of which can be systematically obtained from official documentation.

whereas Chart2Code53 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.

Plotting content diversity Additionally, due to we take more plotting resource into consideration, Chart2Code53 includes more API combinations than exisiting datasets, which indicates Chart2Code53 exhibits much higher content diversity. Further details are shown in Appendix.A.1.

3 Experiments

3.1 Experimental setup

Evaluation Benchmarks To demonstrate the effectiveness of our dataset, we evaluate two mainstream Chart2Code benchmarks: ChartMimic and Plot2Code. We test our model under the direct generation setting, where models generate plotting code directly from given charts. *For ChartMimic benchmark*, we evaluate on its testmini split (containing 600 diverse charts) as it achieves performance comparable to the full setting. The benchmark combines low-level metrics (automatically computed from code similarity across text, layout, type, and color dimensions, averaged as the final score) and high-level GPT-4o-based image comparison scores, with their average as the final metric. Failed code runs get 0 points. We follow these rules exactly. *For Plot2Code benchmark*, we follow ChartCoder (Zhao et al., 2025) and test on its matplotlib split (132 samples). The benchmark evaluates both text-match (measuring text similarity between generated and reference images) and GPT-4o scoring. We report only the GPT-4o metric in our evaluation.

Baselines (1) closed-source MLLMs (Gemini Pro (Team et al., 2025), Claude 3 Opus, GPT-4o (OpenAI et al., 2024)) with strong Chart2Code capabilities; (2) chart-specific MLLMs - TinyChart (Zhang et al., 2024a) (fine-tuned from TinyLLaVA (Zhou et al., 2024) using mixed Chart2Code data included in ChartLlama dataset) and ChartMOE (Xu et al., 2025) ; (3) open-source multimodal LLMs (Qwen2-VL (Wang et al., 2024) and InternVL2 (Chen et al., 2024) families across different model parameters); (4) Chart2Code-specific models: we compare Qwen2-VL-7B fine-tuned on ChartCoder (Zhao et al., 2025) (without Snippet-of-Thought) versus finetuned on our dataset.

Implementation details We conduct fine-tuning experiments on two model families of different parameters: Qwen2-VL-2B, Qwen2-VL-7B, and InternVL2-4B using our Chart2Code53 dataset, with additional comparative experiments performed on Qwen2-VL-7B using the Chartcoder dataset. For the Qwen2-VL series, we implement fine-tuning via the LLaMA-Factory (Zheng et al., 2024) framework, while the InternVL2 models are fine-tuned using their official codebase. We maintain identical training settings across all experiments: the visual encoder remains frozen while other parameters are updated. We use LoRA (Hu et al., 2021) finetuning on A100 GPUs with a global batch size of 16 and a lora_r of 64. We train 2 epochs to ensure full convergence.

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 are trained on a larger-scale Chart2Code dataset and demonstrate 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 achieves SOTA performance. (1) As shown in Table 2, the Qwen2-VL-7B model, after fine-tuning on our dataset, achieves a significant performance improvement. It outperforms other open-source models of much larger parameters, achieving SOTA performance. This strongly validates the effectiveness of our dataset.

(2) On the high-level metric of ChartMimic, the model performs closer to the InternVL2-Llama3-76B, despite having a lower code execution success rate. We believe that this performance gap is more likely due to the inherent limitations in code generation capabilities of the relatively small parameter base model itself.

(3) Furthermore, fine-tuning both Qwen2-VL-2B and InternVL2-4B with our dataset yields consistent performance gains, demonstrating the dataset's effectiveness across model families and varying parameter scales.

(4) Our fine-tuned model on the dataset outper-

Model Name	Params	ChartMimic Execute Rate	ChartMimic Low-Level	ChartMimic High-Level	ChartMimic Overall	Plot2Code Execute Rate	Plot2Code GPT-4o Rating
Close-source Multimodal Large Language Models							
GeminiProVision	-	68.2	53.8	53.3	53.6	68.2	3.7
Claude-3-opus	-	83.3	60.5	60.1	60.3	84.1	3.8
GPT-4o	-	**93.2**	**79.0**	**83.5**	**81.3**	**88.6**	**5.7**
Chart-specific Multimodal Large Language Models							
TinyChart	3.0B	42.5	26.3	25.9	26.1	43.2	2.2
ChartMOE	7.0B	52.7	25.3	22.9	24.1	65.2	2.2
Open-source Multimodal Large Language Models							
Qwen2-VL-2B	3.2B	51.0	22.2	20.1	21.2	52.0	2.4
Qwen2-VL-7B	8.2B	67.0	32.9	35.0	34.0	68.2	3.1
Qwen2-VL-72B	73.2B	73.3	54.4	50.9	52.3	72.0	**4.3**
InternVL2-4B	4.2B	50.5	33.8	38.4	36.1	66.3	2.5
InternVL2-26B	26.0B	69.3	41.4	47.4	44.4	81.3	3.4
InternVL2-Llama3-76B	76.0B	**83.2**	**54.8**	**62.2**	**58.5**	**83.2**	3.9
Chart2Code-specific models							
Qwen2-VL-2B-FT (Chart2Code53)	3.2B	61.0	50.9	48.3	49.6	70.0	3.2
InternVL2-4B-FT (Chart2Code53)	4.2B	78.3	63.4	60.4	61.9	84.8	4.5
Qwen2-VL-7B-FT (Chart2Code53)	8.2B	82.0	68.8	**68.8**	**68.8**	**83.3**	**5.2**
Qwen2-VL-7B-FT (ChartCoder)	8.2B	**86.0**	**69.1**	68.2	68.7	77.3	3.8

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 few parameters, the model fine-tuned on Chart2Code53 achieves state-of-the-art performance across the evaluated benchmarks. Note that we do not include the snip-of-thought method in ChartCoder to make a fair comparison. 'FT' means using the dataset to finetune the model.

formed the results of the same base model fine-tuned on ChartCoder, with only slightly lower low-level metrics on ChartMimic, ***even though our model's execution success rate is significantly lower than that of the ChartCoder model.*** As indicated by the * in Table 3, when we relaxed the evaluation metrics and tested the metrics before the generated code threw exceptions, the experiments show that our model surpass the ChartCoder model on all dimensions of ChartMimic except Layout. As indicated by the ** in Table 3, when we adopt the 'no_filter' setting of ChartMimic and calculate the average metric of multiple samplings, the results also hold.

The model shows consistent significant performance improvements across different categories. As shown in the Figure 4, 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.

3.3 Analysis

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

Model performance consistently improves when increasing code complexity. To evaluate how code complexity affects model performance, we stratified the data by complexity level (measured by code length) for each chart type. Specifically, we fine-tune the Qwen2-VL-2B on four subsets of the dataset of increasing complexity and evaluate performance on ChartMimic low-level score. The results are shown in Figure 5. The results demonstrate a clear positive correlation between code complexity and model performance, with average scores increasing from 35.3 (25% simplest samples) to 50.9 (full dataset). The results demonstrate a positive association between code complexity in training data and model performance.

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	42.1	18.6	16.1	25.3
InternVL2-4B-FT (Chart2Code53)	61.6	74.9	62.9	54.0	63.4
Qwen2-VL-2B-FT (Chart2Code53)	67.3	83.6	67.1	58.4	69.1
Qwen2-VL-7B-FT (ChartCoder)	67.3	**83.6**	**67.1**	58.4	**69.1**
Qwen2-VL-7B-FT (Chart2Code53)	**68.6**	80.7	66.1	**60.2**	68.8
Qwen2-VL-7B-FT (ChartCoder*)	76.5	**96.0**	80.2	68.5	80.3
Qwen2-VL-7B-FT (Chart2Code53*)	**78.5**	95.0	**83.0**	**73.2**	**82.4**
Qwen2-VL-7B-FT (ChartCoder**)	69.3	**93.6**	75.9	63.7	75.6
Qwen2-VL-7B-FT (Chart2Code53**)	**74.0**	93.5	**77.9**	**65.8**	**77.8**

Table 3: Model performance across different dimensions in ChartMimic. * denotes the metrics corresponding to the code executed up to the point before the exception is thrown. ** denotes the 'no_filter' setting of ChartMimic and average metric of multiple samplings. 'FT' means using the dataset to finetune the model.

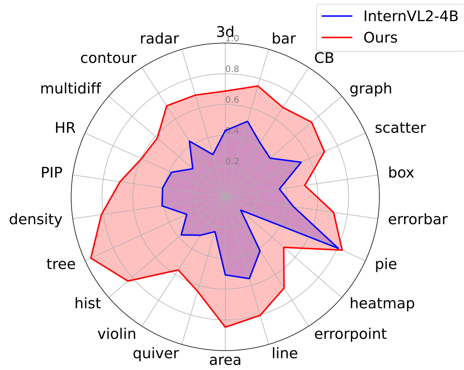

Figure 4: Performance across all chart types in ChartMimic. Our model shows consistent improvement across all chart types.

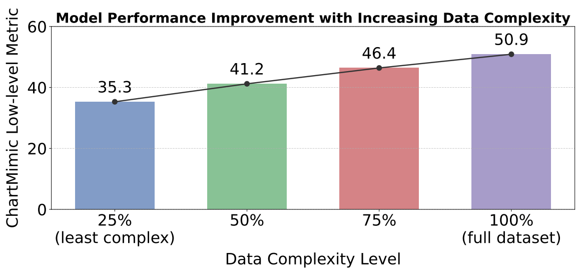

Figure 5: Model performance consistently improves when increasing code complexity.

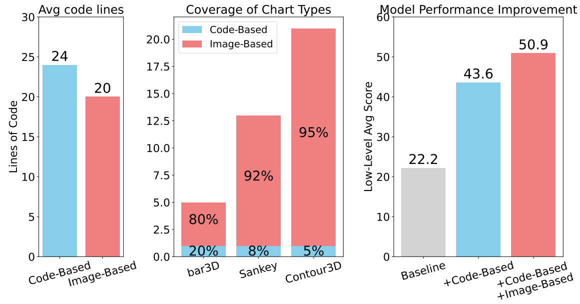

Figure 6: Relative contributions of each synthesis pipeline.

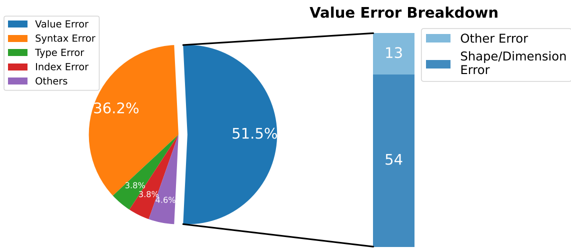

Figure 7: Error distributions of our model.

Both synthesis pipelines contribute to statistics and performance. We conduct a comprehensive analysis of both image-based and code-based data generation pipelines from statistical and performance perspectives. *From statistical perspective*, code-based synthesis yields slightly higher chart complexity (avg. 24 lines of code per chart vs. 20 for image-based) as shown in the left panel of Figure 6. Image-based synthesis improves coverage of sparse categories in Code-based synthesis (Because users seldom open-source their code of some chart types, such as contour3D chart and sankey chart) as shown in the middle panel of Figure 6. *From performance perspective*, we finetune Qwen2-VL-2B on the code-based data first and then add image-based data. As shown in the right panel of Figure 6, both data pipelines contribute to model improvement.

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 errors of the model are Syntax errors and variable planning errors. As shown in the Figure 7, coding errors are primarily syntax and value errors, with the latter mainly due to dimension mismatches of the variables defined before they are used. This indicates that apart from general coding abilities, variable planning is an important ability for the Chart2Code task that might be considered to be further improved, which may be challenging due to the auto-regressive nature of current MLLMs.

3.4 Case study

We present in Figure 8 a qualitative analysis of the Qwen2-VL-7B model under three settings: (1) the plain model. (2) ChartCoder-tuned model, and (3) Chart2Code53-tuned model. Each image is generated by executing the models prediction code given the gold chart image.

The first two rows show that the plain model fails to generate more complex composite charts. The ChartCoder-tuned model correctly identifies the chart types but fails to combine them effectively. In contrast, the Chart2Code53-tuned model reconstructs such charts more accurately.

The third row shows that our model accurately captures the color gradient in the gold reference

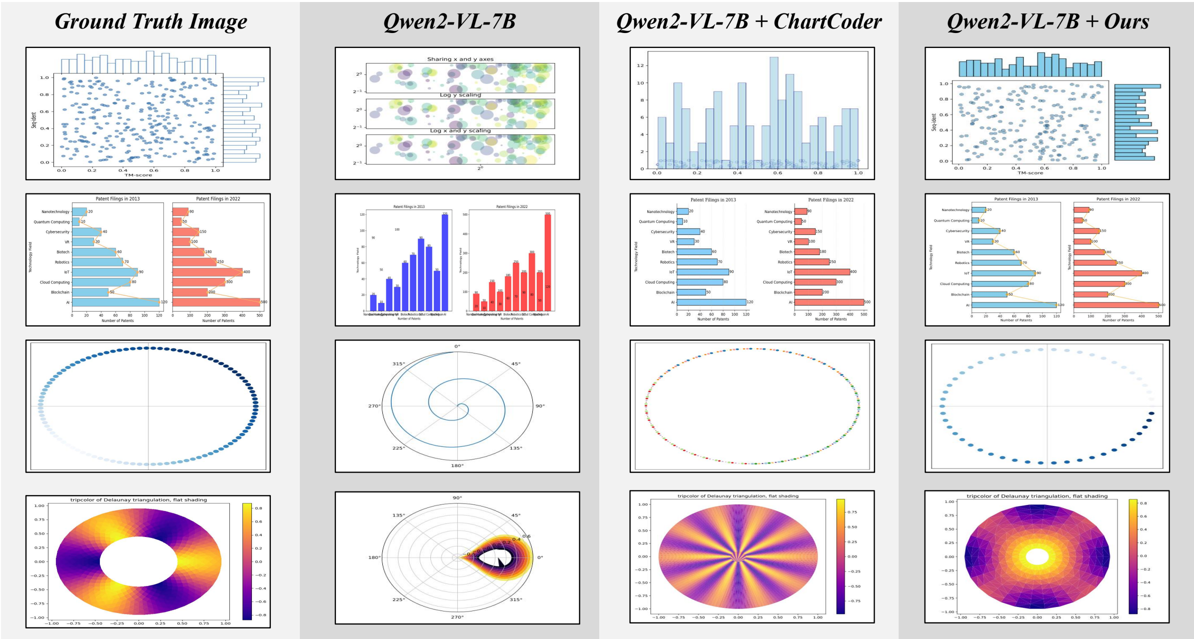

Figure 8: Qualitative examples of images generated by executing code from different models

chart. The fourth row demonstrates that the Chart2Code53-tuned model can detect and reproduce hollow circles in the gold image. Compared with the plain and the ChartCoder-tuned models, the Chart2Code53-tuned model effectively handles more diverse chart styling designs.

In summary, our diverse and complexity-varied Chart2Code53 dataset significantly enhances the model's Chart2Code capability.

4 Related work

Chart understanding Recent Chart understanding works primarily build upon MLLMs. ChartAssistant (Meng et al., 2024), ChartLlama (Han et al., 2023), and TinyChart (Zhang et al., 2024a) directly fine-tune existing MLLMs. ChartMOE (Xu et al., 2025) employs a Mixture-of-Experts architecture to integrate three alignment tasks (chart-to-text, chart-to-json, and chart-to-code), proving that chart-to-code tasks significantly enhance chart understanding. However, our experiments reveal that these chart-specific models still exhibit poor Chart2Code capability. Our work specifically focuses on improving MLLMs' Chart2Code performance.

Multimodal code generation Multimodal code generation refers to producing source code using both non-textual modalities and pure textual information, where the generated code serves as the final output. Existing works can be categorized into three groups: (1) Visual Programming: Benchmarks such as MMCode (Li et al., 2024a) and HumanEval-V (Zhang et al., 2025) evaluate code generation from multimodal inputs (images + text). (2) Front-end code generation: Design2Code (Si et al., 2025) provides real-world websites as a benchmark, while Web2Code (Yun et al., 2024) offers a larger-scale alternative. (3) Chart-to-code generation: The task requires MLLMs to accurately interpret charts and generate corresponding code. Existing benchmarks include Plot2Code (Wu et al., 2025) and ChartMimic (Shi et al., 2025), revealing significant performance gaps in current open-source models ChartLlama/ChartAssistant use text LLMs to synthesize code from specified chart types/styles, suffering from limited diversity. Recent approaches ReachQA (He et al., 2024) (using evol-instruct) and ChartMOE (using self-instruct) improve complexity but remain constrained by scale and diversity. Our work introduces a dual-pipeline for data synthesis and constructs Chart2Code53, addressing three key limitations: (1) limited scale (2) limited diversity, (3) limited complexity.

5 Conclusion

This paper addresses the limitations of existing Chart2Code-related datasets, including insufficient quantity, diversity, and complexity. We propose a dual data synthesis pipeline to create a large-scale Chart2Code training dataset and conduct fine-tuning experiments on open-source models. The results show that the model achieves SOTA performance with fewer parameters. We hope our dataset and analysis 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.

Acknowledgments

We gratefully acknowledge the support of the National Key R&D Program Project (2024YFE0203700) and the National Natural Science Foundation of China (NSFC) via grant 62236004, 62206078, 62441603 and 62476073.

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.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

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.

Kaixin Li, Yuchen Tian, Qisheng Hu, Ziyang Luo, Zhiyong Huang, and Jing Ma. 2024a. Mmcode: Benchmarking multimodal large language models for code generation with visually rich programming problems. *Preprint*, arXiv:2404.09486.

Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. 2024b. 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.

Sachit Menon and Carl Vondrick. 2023. Visual classification via description from large language models. In *The Eleventh International Conference on Learning Representations*.

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*.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2Code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3956–3974, Albuquerque, New Mexico. Association for Computational Linguistics.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Jack Krawczyk, Cosmo Du, Ed Chi, Heng-Tze Cheng, Eric Ni, Purvi Shah, Patrick

Kane, Betty Chan, Manaal Faruqui, Aliaksei Severyn, Hanzhao Lin, YaGuang Li, Yong Cheng, Abe Ittycheriah, Mahdis Mahdieh, Mia Chen, Pei Sun, Dustin Tran, Sumit Bagri, Balaji Lakshminarayanan, Jeremiah Liu, Andras Orban, Fabian Güra, Hao Zhou, Xinying Song, Aurelien Boffy, Harish Ganapathy, Steven Zheng, HyunJeong Choe, Ágoston Weisz, Tao Zhu, Yifeng Lu, Siddharth Gopal, Jarrod Kahn, Maciej Kula, Jeff Pitman, Rushin Shah, Emanuel Taropa, Majd Al Merey, Martin Baeuml, Zhifeng Chen, Laurent El Shafey, Yujing Zhang, Olcan Sercinoglu, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Gaurav Singh Tomar, Evan Senter, Martin Chadwick, Ilya Kornakov, Nithya Attaluri, Iñaki Iturrate, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Xavier Garcia, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Ravi Addanki, Antoine Miech, Annie Louis, Denis Teplyashin, Geoff Brown, Elliot Catt, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogoziska, Vitaliy Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lui, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çalar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer,

Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakievi, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Sidharth Mudgal, Romina Stella, Kevin Brooks, Gautam Vasudevan, Chenxi Liu, Mainak Chain, Nivedita Melinkeri, Aaron Cohen, Venus Wang, Kristie Seymore, Sergey Zubkov, Rahul Goel, Summer Yue, Sai Krishnakumaran, Brian Albert, Nate Hurley, Motoki Sano, Anhad Mohananey, Jonah Joughin, Egor Filonov, Tomasz Kpa, Yomna Eldawy, Jiawern Lim, Rahul Rishi, Shirin Badiezadegan, Taylor Bos, Jerry Chang, Sanil Jain, Sri Gayatri Sundara Padmanabhan, Subha Puttagunta, Kalpesh Krishna, Leslie Baker, Norbert Kalb, Vamsi Bedapudi, Adam Kurzrok, Shuntong Lei, Anthony Yu, Oren Litvin, Xiang Zhou, Zhichun Wu, Sam Sobell, Andrea Siciliano, Alan Papir, Robby Neale, Jonas Bragagnolo, Tej Toor, Tina Chen, Valentin Anklin, Feiran Wang, Richie Feng, Milad Gholami, Kevin Ling, Lijuan Liu, Jules Walter, Hamid Moghaddam, Arun Kishore, Jakub Adamek, Tyler Mercado, Jonathan Mallinson, Siddhinita Wandekar, Stephen Cagle, Eran Ofek, Guillermo Garrido, Clemens Lombriser, Maksim Mukha, Botu Sun, Hafeezul Rahman Mohammad, Josip Matak, Yadi Qian, Vikas Peswani, Pawel Janus, Quan Yuan, Leif Schelin, Oana David, Ankur Garg, Yifan He, Oleksii Duzhyi, Anton Älgmyr, Timothée Lottaz, Qi Li, Vikas Yadav, Luyao Xu, Alex Chinien, Rakesh Shivanna, Aleksandr Chuklin, Josie Li, Carrie Spadine, Travis Wolfe, Kareem Mohamed, Subhabrata Das, Zihang Dai, Kyle He, Daniel von Dincklage, Shyam Upadhyay, Akanksha Maurya, Luyan Chi, Sebastian Krause, Khalid Salama, Pam G Rabinovitch, Pavan Kumar Reddy M, Aarush Selvan, Mikhail Dektiarev, Golnaz Ghiasi, Erdem Guven, Himanshu Gupta, Boyi Liu, Deepak Sharma, Idan Heimlich Shtacher, Shachi Paul, Oscar Akerlund, François-Xavier Aubet, Terry Huang, Chen Zhu, Eric Zhu, Elico Teixeira, Matthew Fritze, Francesco Bertolini, Liana-Eleonora Marinescu, Martin Bölle, Dominik Paulus, Khyatti Gupta, Tejasi Latkar, Max Chang, Jason Sanders, Roopa Wilson, Xuewei Wu, Yi-Xuan Tan, Lam Nguyen Thiet, Tulsee Doshi, Sid Lall, Swaroop Mishra, Wanming Chen, Thang Luong, Seth Benjamin, Jasmine Lee, Ewa Andrejczuk, Dominik Rabiej, Vipul Ranjan, Krzysztof Styrc, Pengcheng Yin, Jon Simon, Malcolm Rose Harriott, Mudit Bansal, Alexei Robsky, Geoff Bacon, David Greene, Daniil Mirylenka, Chen Zhou, Obaid Sarvana, Abhimanyu Goyal, Samuel Andermatt, Patrick Siegler, Ben Horn, Assaf Israel, Francesco Pongetti, Chih-Wei "Louis" Chen, Marco Selvatici, Pedro Silva, Kathie Wang, Jackson Tolins, Kelvin Guu, Roey Yogev, Xiaochen Cai, Alessandro Agostini, Maulik Shah, Hung Nguyen, Noah Ó Donnaile, Sébastien Pereira, Linda Friso, Adam Stambler, Adam Kurzrok, Chenkai Kuang, Yan Romanikhin, Mark Geller, ZJ Yan, Kane Jang, Cheng-Chun Lee, Wojciech Fica, Eric Malmi, Qijun Tan, Dan Banica, Daniel Balle, Ryan Pham, Yanping Huang, Diana Avram, Hongzhi Shi, Jasjot Singh, Chris Hidey, Niharika Ahuja, Pranab Saxena, Dan Dooley, Srividya Pranavi Potharaju, Eileen O'Neill, Anand Gokulchandran, Ryan Foley, Kai Zhao, Mike Dusenberry, Yuan Liu, Pulkit Mehta, Ragha Kotikalapudi, Chalence Safranek-Shrader, Andrew Goodman, Joshua Kessinger, Eran Globen, Prateek Kolhar, Chris Gorgolewski, Ali Ibrahim, Yang Song, Ali Eichenbaum, Thomas Brovelli, Sahitya Potluri, Preethi Lahoti, Cip Baetu, Ali Ghorbani, Charles Chen, Andy Crawford, Shalini Pal, Mukund Sridhar, Petru Gurita, Asier Mujika, Igor Petrovski, Pierre-Louis Cedoz, Chenmei Li, Shiyuan Chen, Niccolò Dal Santo, Siddharth Goyal, Jitesh Punjabi, Karthik Kappaganthu, Chester Kwak, Pallavi LV, Sarmishta Velury, Himadri Choudhury, Jamie Hall, Premal Shah, Ricardo Figueira, Matt Thomas, Minjie Lu, Ting Zhou, Chintu Kumar, Thomas Jurdi, Sharat Chikkerur, Yenai Ma, Adams Yu, Soo Kwak, Victor Ähdel, Sujeevan Rajayogam, Travis Choma, Fei Liu, Aditya Barua, Colin Ji, Ji Ho Park, Vincent Hellendoorn, Alex Bailey, Taylan Bilal, Huanjie Zhou, Mehrdad Khatir, Charles Sutton, Wojciech Rzadkowski, Fiona Macintosh, Roopali Vij, Konstantin Shagin, Paul Medina, Chen Liang, Jinjing Zhou, Pararth Shah, Yingying Bi, Attila Dankovics, Shipra Banga, Sabine Lehmann, Marissa Bredesen, Zifan Lin, John Eric Hoffmann, Jonathan Lai, Raynald Chung, Kai Yang, Nihal Balani, Arthur Brainskas, Andrei Sozanschi, Matthew Hayes, Héctor Fernández Alcalde, Peter Makarov, Will Chen, Antonio Stella, Liselotte Snijders, Michael Mandl, Ante Kärrman, Pawe Nowak, Xinyi Wu, Alex Dyck, Krishnan Vaidyanathan, Raghavender R, Jessica Mallet, Mitch Rudominer, Eric Johnston, Sushil Mittal, Akhil Udathu, Janara Christensen, Vishal Verma, Zach Irving, Andreas Santucci, Gamaleldin Elsayed, Elnaz Davoodi, Marin Georgiev, Ian Tenney, Nan Hua, Geoffrey Cideron, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Dylan Scandinaro, Heinrich Jiang, Jasper Snoek, Mukund Sundararajan, Xuezhi Wang, Zack Ontiveros, Itay Karo, Jeremy Cole, Vinu Rajashekhar, Lara Tumeh, Eyal Ben-David, Rishub Jain, Jonathan Uesato, Romina Datta, Oskar Bunyan, Shimu Wu, John Zhang, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Jane Park, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Lu-

owei Zhou, Jonathan Evens, William Isaac, Geoffrey Irving, Edward Loper, Michael Fink, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Ivan Petrychenko, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Evan Palmer, Paul Suganthan, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikoaj Rybiski, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Ginger Perng, Elena Allica Abellan, Mingyang Zhang, Ishita Dasgupta, Nate Kushman, Ivo Penchev, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Daniel Andor, Pedro Valenzuela, Minnie Lui, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Ken Franko, Anna Bulanova, Rémi Leblond, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Mark Omernick, Colton Bishop, Rachel Sterneck, Rohan Jain, Jiawei Xia, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Daniel J. Mankowitz, Alex Polozov, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Matthieu Geist, Ser tan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Kathy Wu, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Saaber Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Yeqing Li, Nir Levine, Ariel Stolovich, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Charlie Deck, Hyo Lee, Zonglin Li, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Sho Arora, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Pluciska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Lynette Webb, Sahil Dua, Dong Li, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Evgenii Eltyshev, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Christof Angermueller, Xiaowei Li, Anoop Sinha, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Denny Zhou, Komal Jalan, Dinghua Li, Blake Hechtman, Parker Schuh, Milad Nasr, Kieran Milan, Vladimir Mikulik, Juliana Franco, Tim Green, Nam Nguyen, Joe Kelley, Aroma Mahendru, Andrea Hu, Joshua Howland, Ben Vargas, Jeffrey Hui, Kshitij Bansal, Vikram Rao, Rakesh Ghiya, Emma Wang, Ke Ye, Jean Michel Sarr, Melanie Moranski Preston, Madeleine Elish, Steve Li, Aakash Kaku, Jigar Gupta, Ice Pasupat, Da-Cheng Juan, Milan Someswar, Tejvi M., Xinyun Chen, Aida Amini, Alex Fabrikant, Eric Chu, Xuanyi Dong,

Amruta Muthal, Senaka Buthpitiya, Sarthak Jauhari, Nan Hua, Urvashi Khandelwal, Ayal Hitron, Jie Ren, Larissa Rinaldi, Shahar Drath, Avigail Dabush, Nan-Jiang Jiang, Harshal Godhia, Uli Sachs, Anthony Chen, Yicheng Fan, Hagai Taitelbaum, Hila Noga, Zhuyun Dai, James Wang, Chen Liang, Jenny Hamer, Chun-Sung Ferng, Chenel Elkind, Aviel Atias, Paulina Lee, Vít Listík, Mathias Carlen, Jan van de Kerkhof, Marcin Pikus, Krunoslav Zaher, Paul Müller, Sasha Zykova, Richard Stefanec, Vitaly Gatsko, Christoph Hirnschall, Ashwin Sethi, Xingyu Federico Xu, Chetan Ahuja, Beth Tsai, Anca Stefanoiu, Bo Feng, Keshav Dhandhania, Manish Katyal, Akshay Gupta, Atharva Parulekar, Divya Pitta, Jing Zhao, Vivaan Bhatia, Yashodha Bhavnani, Omar Alhadlaq, Xiaolin Li, Peter Danenberg, Dennis Tu, Alex Pine, Vera Filippova, Abhipso Ghosh, Ben Limonchik, Bhargava Urala, Chaitanya Krishna Lanka, Derik Clive, Yi Sun, Edward Li, Hao Wu, Kevin Hongtongsak, Ianna Li, Kalind Thakkar, Kuanysh Omarov, Kushal Majmundar, Michael Alverson, Michael Kucharski, Mohak Patel, Mudit Jain, Maksim Zabelin, Paolo Pelagatti, Rohan Kohli, Saurabh Kumar, Joseph Kim, Swetha Sankar, Vineet Shah, Lakshmi Ramachandruni, Xiangkai Zeng, Ben Bariach, Laura Weidinger, Tu Vu, Alek Andreev, Antoine He, Kevin Hui, Sheleem Kashem, Amar Subramanya, Sissie Hsiao, Demis Hassabis, Koray Kavukcuoglu, Adam Sadovsky, Quoc Le, Trevor Strohman, Yonghui Wu, Slav Petrov, Jeffrey Dean, and Oriol Vinyals. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

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. *Preprint*, arXiv:2409.12191.

Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. 2025. Plot2Code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3006–3028, Albuquerque, New Mexico. Association for Computational Linguistics.

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*.

Yue Yang, Ajay Patel, Matt Deitke, Tanmay Gupta, Luca Weihs, Andrew Head, Mark Yatskar, Chris Callison-Burch, Ranjay Krishna, Aniruddha Kembhavi, and Christopher Clark. 2025. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. *Preprint*, arXiv:2502.14846.

Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Qazim Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, Timothy Baldwin, Zhengzhong Liu, Eric P. Xing, Xiaodan Liang, and Zhiqiang Shen. 2024. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. *Preprint*, arXiv:2406.20098.

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.

Fengji Zhang, Linquan Wu, Huiyu Bai, Guancheng Lin, Xiao Li, Xiao Yu, Yue Wang, Bei Chen, and Jacky Keung. 2025. Humaneval-v: Benchmarking high-level visual reasoning with complex diagrams in coding tasks. *Preprint*, arXiv:2410.12381.

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.

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. *Preprint*, arXiv:2501.06598.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *Preprint*, arXiv:2403.13372.

Baichuan Zhou, Ying Hu, Xi Weng, Junlong Jia, Jie Luo, Xien Liu, Ji Wu, and Lei Huang. 2024. Tinyllava: A framework of small-scale large multimodal models. *Preprint*, arXiv:2402.14289.

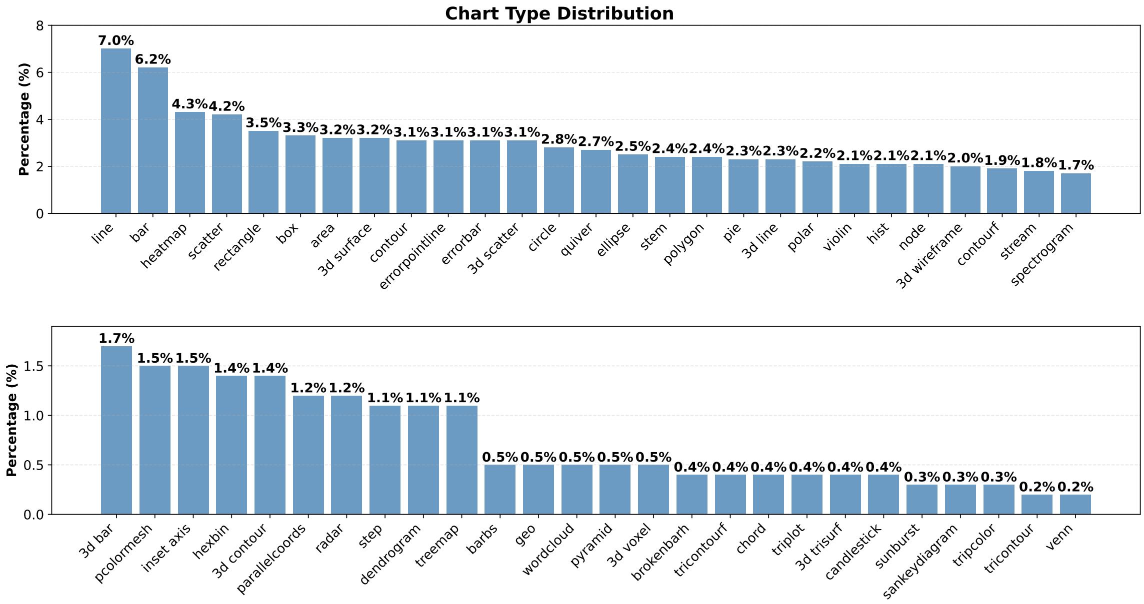

Figure 9: Type distribution of Chart2Code53. The upper panel displays the top half of types by prevalence, while the lower panel displays the remaining types.

A Appendix

A.1 Statistics

Chart Type distribution The type distribution is shown in Figure 9. As illustrated, the distribution of categories in our dataset is well-balanced.

Plotting Content diversity To evaluate the diversity of our dataset, we employ two metrics: API Combination and Average Distinct n-gram.

- **API Combination:** For a single code snippet, its API Combination is defined as the multiset of all API names used in that snippet. Across the training set, the number of distinct API Combinations reflects the variety of multisets derived from all code snippets. This metric corresponds directly to diverse visualization intents and plotting patterns, inherently capturing the richness of users' programming logic. As shown in Table 1, our dataset demonstrates substantial diversity in plotting logic.

- **Average Distinct n-gram:** This metric calculates the average number of distinct n-grams (for n=1 to 5) across all samples in the dataset. By considering the entire code text, it better reflects the diversity of data and parameter definitions. Results presented in table 4 confirm that our dataset exhibits strong diversity in both data and parameters.

Dataset name	# Data samples	# Avg distinct n-grams
Text2Chart31	11K	365K
ReachQA	3K	306K
CoSyn	53K	1892K
Chart2Code53	130K	7205K

Table 4: Average distinct n-gram metric.

A.2 Quality control details

Since our data synthesis pipeline generates code and then uses it to produce corresponding images, an inherent correspondence exists between the images and the code. To ensure quality control, we verify both the aesthetic quality of the images and the absence of redundant code segments unrelated to plotting. For code quality, we focus on identifying invalid code segments that do not visibly affect the final rendered image. Manual inspection of 2K samples reveals the following recurring issues[4]:

- Overridden statements (setting plt.axis(False) after using ax.xsticks()).

- The iterable variable (a numpy array) is only partially visualized in the generated plot.

- Redundant if-condition branches.

[4]We don't use GPT-4o to check the code as we found that GPT-4o struggles to accurately identify these issues based solely on given chart images and code, and frequently flags non-existent problems.

These errors likely originate from the inclusion of user debugging logic in the original web-sourced code snippets. While less impactful for Chart2Code, such issues could adversely affect downstream chart comprehension tasks. Through random sampling, we estimated the prevalence of such problematic samples to be acceptably low (less than 3.2% of the total data). The entire dataset verification process was conducted independently by the first author to ensure consistency.

A.3 Qualitative samples of synthesised charts

In this section, we show some qualitative samples of our synthesised dataset examples.

Figure 10: Synthesised chart example. This figure presents a silhouette analysis for KMeans clustering on sample data with five clusters. The left panel shows the silhouette plot, where each cluster is represented by a distinct color. The right panel visualizes the clustered data in a two-dimensional feature space, with each cluster labeled and colored differently. It's a combination of scatter chart,axline chart and fillbetween (area) chart with text.

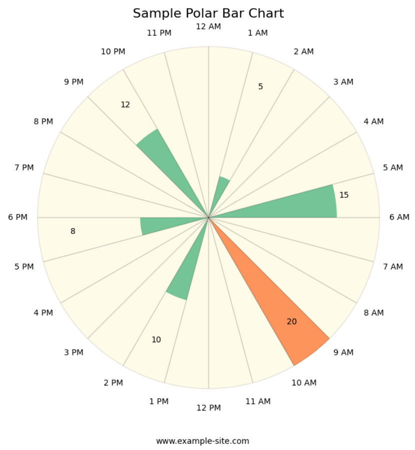

Figure 11: Synthesised chart example. This chart illustrates the distribution of a specific variable across different time intervals within a 24-hour period. Each segment represents an hour of the day, and the length of the bar within each segment indicates the magnitude of the variable being measured. It's a Polar bar chart.

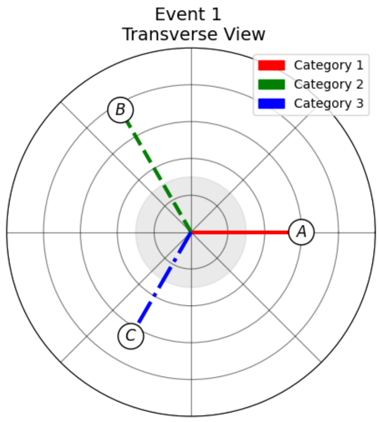

Figure 12: Synthesised chart example. This transverse view chart visualizes the spatial distribution of three different categories (Category 1, Category 2, and Category 3) across a radial plane. Each category is represented by a distinct color: red for Category 1, green for Category 2, and blue for Category 3. Points A, B, and C indicate specific locations where each category is observed. It's a combination of line chart and scatter chart in polar axis.

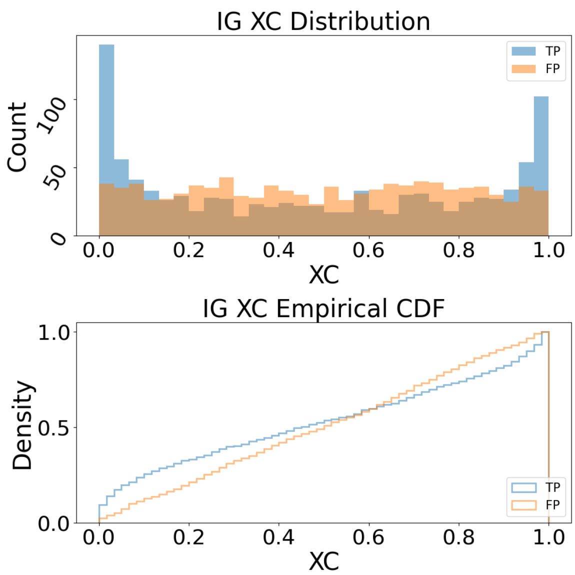

Figure 13: Synthesised chart example. This figure illustrates the IG XC distribution and empirical CDF, where the top histogram shows the counts of true positives (TP) and false positives (FP) across different XC values, and the bottom plot displays their cumulative density functions. It's a combination of hist chart and density chart.

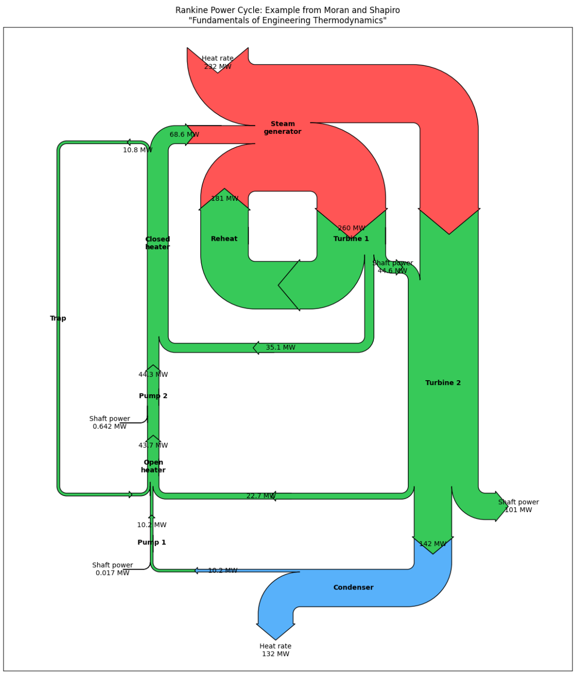

Figure 14: Synthesised chart example. This diagram illustrates a Rankine power cycle, a thermodynamic cycle commonly used in power plants for converting heat into mechanical work.The diagram highlights the flow of the working fluid through these stages, emphasizing the transformation of energy forms throughout the process. It's Sankey chart.

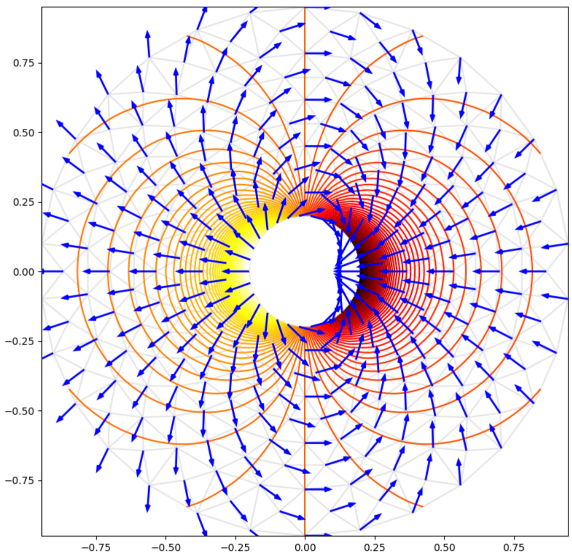

Figure 15: Synthesised chart example. This figure illustrates a vector field plot, depicting the flow and magnitude of vectors in a two-dimensional space. The color gradient from yellow to red represents varying magnitudes, with yellow indicating lower values and red indicating higher values at the center. The vectors, represented by arrows, show the direction of the flow, converging towards the center. It's a quiver chart.

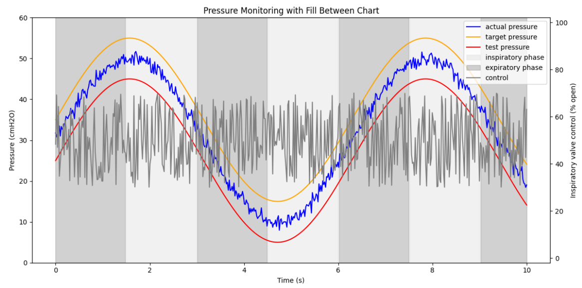

Figure 16: Synthesised chart example. The chart provided illustrates the pressure waveform with PEEP (Positive End-Expiratory Pressure) during mechanical ventilation. The blue line represents actual pressure, while the orange line indicates target pressure, and the red line denotes tidal pressure. The shaded grey regions indicate the inspiratory and expiratory phases of the breathing cycle, with the expiratory phase marked by the grey background. It's a line chart with a varying background.

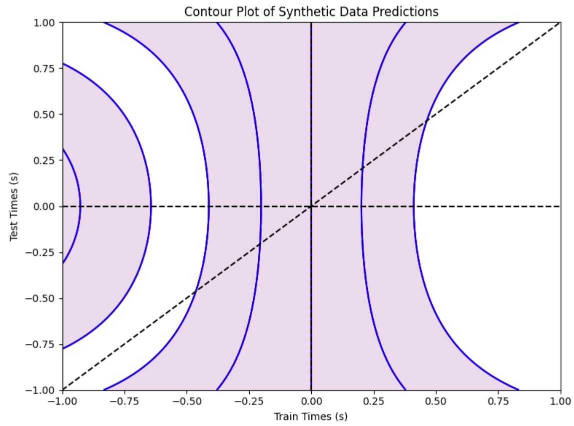

Figure 17: Synthesised chart example. This contour plot illustrates synthetic data predictions across a two-dimensional parameter space. The plot features contour lines that represent levels of constant predicted values, with shaded regions indicating areas of similar prediction magnitudes. The diagonal dashed line signifies a reference or baseline condition. It's contour and line chart.