
A DETAILS OF THE DATA GENERATION PIPELINE

A.1 DATASET FILTERING

In Sec. 3.1, we introduce a novel data generation pipeline that leverages text-only LLMs. This pipeline enables us to collect chart images along with various data and QA pairs without extensive human effort, thereby reducing the cost of creating pairwise data. However, LLMs are not perfect and can make mistakes in either data generation or code script generation. Thus, in this section, we discuss the data filtering techniques we use to improve the quality of the synthetic dataset. The generation pipeline is split into three parts: shared template generation, data and QA generation, and code script generation. We now detail the filtering process for each part.

Shared template and README The shared template and README file for each chart type form the core of the entire data generation process, as the subsequent raw data, QA, and Python script are based on the shared template. Therefore, for the shared template, we deploy a human check to ensure the template contains necessary elements for the chart (i.e., title, x-axis, y-axis, data). Additionally, humans are required to verify the correctness of the chart type definitions in the README. Note that we consider 20 different chart types; thus, there are 20 template JSON files along with the READMEs in our dataset.

Data and QA generation Since all the data should follow the template JSON, for the data generation part, we apply filtering based on the JSON structure. Specifically, we remove generated data that deviates from the template file by comparing the elements in the keys of the JSON dictionary and the data types of all the values. As for QA generation, we check the structure of the output dictionary. In detail, the keys of the output QA dictionary should contain summary, description, literal, inferential, and reasoning QAs. We filter out QAs with missing attributes.

Code script generation We predefined a code expert GPT-4 to use four different Python libraries to plot the chart images: Matplotlib, Plotly, Pygal, and Seaborn. The advantage of these libraries is that if the Python code or input data is incorrect in terms of structure or other errors, the generated image will either be missing with a Python error or display a "No Data" icon. Thus, we apply a two-step filtering process: (1) Python Error Filtering: If there is an error while running the Python script to generate the image, we will remove the script and the corresponding JSON data. (2) OCR Tool Filtering: If the image is generated but there is some other error, the output image will display a "No Data" icon on it. To this end, we further use an OCR tool to detect whether there is any "No Data" icon in the images. If so, we will remove the data and script accordingly.

A.2 DETAILS OF EXPERT GPT-4

In the GPT-4 module, before feeding the prompts for generation, we must specify the system message for GPT-4. These system messages explicitly inform GPT-4 about the environment for the generation and the role it must play in this task, helping the model to output precise responses that match users' needs. Thus, in our data generation pipeline, we encounter three different circumstances, and we have three different system messages for GPT-4, tailoring it to become an expert accordingly. We provide the details of these system message in Table A6 for reference.

B IMPLEMENTATION DETAILS

In this paper, unless otherwise specified, we employ LLaVA-7B as our model architecture, based on the official LLaVA repository.⁴ We use the CLIP ViT large model with a patch size of 14 as the vision encoder, and adopt a 2-layer MLP with GeLU (Hendrycks & Gimpel, 2016) as the activation function.⁵ For the language model, we use Vicuna v1.5 (Zheng et al., 2023) as the LLM backbone. Regarding training details, we follow the default settings in LLaVA v1.5, with a batch size of 32 for pretraining and 16 for finetuning. The model is trained for 1 epoch with learning rates of $2e-4$ and $2e-5$ for pretraining and finetuning, respectively, using a cosine learning rate scheduler, weight

⁴LLaVA: <https://github.com/haotian-liu/LLaVA>.

⁵clip-vit-large: <https://huggingface.co/openai/clip-vit-large-patch14-336>.

Table A6: System messages for GPT-4.

Model	System messages
JSON expert GPT-4	You are an AI chatbot designed to help users generate a JSON template file along with a README for a specific chart type. Once users specify the chart type, you will have to determine the necessary attributes for plotting a chart, including but not limited to the title, x-axis, y-axis, and data. The data part should have a general structure that covers both simple and complex examples for the chart type. As for the README, it should explain the meaning and type of each attribute or label, and also the definition for this chart type. The generated template and README will then be used in creating raw data and a Python script for visualizing the chart.
Data expert GPT-4	You are an expert in generating question and answer pairs based on raw chart data. Your role involves carefully examining chart data, which is presented in JSON format, and creating relevant question and answer pairs. These pairs will be used for instructional tuning of a vision and language model. Along with chart data, you will receive a JSON template and a README file that provides additional information on the meaning of each attribute in the JSON data. It is important to review all the provided materials thoroughly to ensure the question and answer pairs are accurate and useful for model training.
Code expert GPT-4	You are a Python code AI assistant and good at chart image plotting. Now, you are asked to modify the Python code to a general version that can take any JSON data matching the definition in the template.

Table A7: Comparative analysis with existing benchmarks for chart understanding evaluations. * denotes unbounded chart types. Chart variation refers to whether the dataset contains chart images with different styles but sharing the same raw data.

Benchmark	# Image	# Chart type	Avg. # QAs per image	Multi-level QAs per image	Raw data per image	Chart Variation
PlotQA Methani et al. (2020)	33.7k	3	1	✗	✗	✗
ChartQA Masry et al. (2022)	1.5k	3	1	✗	✓	✗
Chart-to-text Kantharaj et al. (2022b)	6.6k	6	1	✗	✗	✗
MMC Liu et al. (2024a)	2k	6	1	✗	✓	✗
Chartbench Xu et al. (2023)	2.1k	9	9	✓	✗	✗
ChartX Xia et al. (2024)	6k	18	1	✗	✓	✗
CharXiv Wang et al. (2024)	2.3k	*	5	✓	✗	✗
Ours	5.48k	20	13.5	✓	✓	✓

decay of 0, and a warmup ratio of 0.03. For computational resources, we use 8 A100 GPUs with bf16 settings for both pretraining and finetuning. Considering the amount of data used, pretraining takes approximately one day and finetuning takes around three days using 8 A100 GPUs.

C DETAILS OF THE GENERATED BENCHMARK

C.1 MOTIVATIONS AND DESIGN PRINCIPLE

To measure the comprehensive chart understanding ability of MLLMs, we assume that models can understand the underlying data of various chart types and perform QAs related to charts. These QAs shouldn't be limited to simple questions about the title or x-axis; instead, they should range from basic to advanced QAs, requiring models to have a global conceptual understanding or even perform mathematical reasoning. However, existing benchmarks either lack a diverse range of chart types or fail to provide comprehensive QAs for each image, lack of a full assessment of understanding from multiple perspectives. To this end, we leverage our powerful data generation pipeline to generate a small set of data and apply filtering through an automatic pipeline and human evaluation. The resulting benchmark has several features to facilitate research in scientific chart understanding: (1) 20 different chart types, covering general use cases to scientific reports; (2) comprehensive QAs for

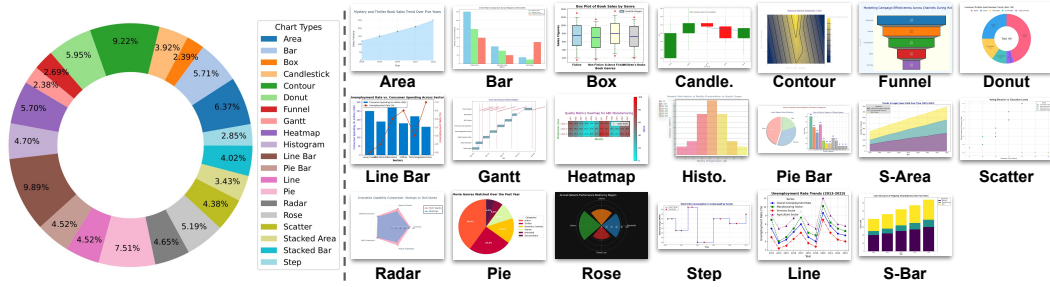


Figure A6: **Overview of our benchmark.** Left: Data distribution across different chart types. Right: Comprehensive list of chart types in the dataset with examples.

each image, including literal, inferential, and reasoning questions, similar to the design of the training data; and (3) raw data for each image, measuring the models' ability to understand the underlying data. We provide a comparison with previous benchmarks in Table A7.

C.2 DATASET STATISTICS

We provide the data statistics after filtering in Table A7. Additionally, a comprehensive list of all the chart types and the data distribution for each type is provided in Fig. A6. Specifically, for each chart type, we initially create $M = 30$ JSON data and $N = 12$ Python scripts. For each JSON dataset, we generate 15 QAs. As mentioned in Sec. 3.1, since all code scripts and data share the same structure, our generated data can be universally applied to any generated code and vice versa. In this way, we obtain approximately 360 unique chart images and around $5.4k$ chart-QA pairs for each chart type. After data filtering, we retain $74k$ chart-QA pairs in total, resulting in an average of $3.5k$ pairs per chart type and an average of 13.5 QAs per image. By applying multiple Python scripts to the same data point, our benchmark includes 608 unique data points, with an average of 9.2 variations per data point.

C.3 DATA FILTERING

The benchmark is generated by our data generation pipeline, while the LLMs are not faultless, resulting in some noisy question-answer pairs. Unlike training data, noisy data can be problematic for benchmark, and cannot accurately access the model performance. Thus, to make sure the quality of the benchmark, we leverage automatic filtering process and human evaluation to filter chart images and question-answer pairs. We now detail the data filtering process of the proposed benchmark.

General data filtering. In the first stage of data filtering, we leverage the automatic data filtering pipeline used for our training data, as mentioned in Appendix A. Specifically, since the JSON data shares the same structure, we can filter out JSON files with incorrect dictionary structures. Furthermore, since the generated QAs follow a predefined dictionary structure, we can filter out QAs with missing attributes or keys. Lastly, when generating chart images using Python code and JSON, we record any execution errors and further filter out Python scripts that produce error messages.

Human evaluation - image filtering. To make sure the quality of the benchmark, we further adopt human evaluation. Recall that, in our benchmark, we have 608 unique data points, with an average of 9.2 variations per data point. To reduce the complexity and cost of human evaluation, we first invite human workers to check the image first, in which human workers have to go through all chart images and check the readability of the chart. Specifically, workers will have to remove chart images if they have missing data points comparing to other variation or they are draw incorrect (potentially due to incorrect python code).

Human evaluation - QA filtering. After filtering the chart images, we then conduct QA filtering. In this step, we ask human workers to review 608 unique data points, with each data point having 15 QAs, resulting in approximately $9k$ test pairs. Specifically, for each test pair, we provide the human

In-context examples	Here are some examples of the evaluation process:
	Example of ignoring unit:
	Question: What is the box office revenue in Week 1?, Groundtruth answer: 20, Predicted answer: 20 million, Output: True
	Example of ignoring unit:
	Question: What is the value of the Alternative Rock music genre?, Groundtruth answer: 5, Predicted answer: 5.0%, Output: True
	Example of ignoring unit:
	Question: What is the total revenue increase from 2013 to 2022?, Groundtruth answer: 250, Predicted answer: Approximately 300 million USD, Output: False
	Example of same meaning:
	Question: Which month had the lowest soil moisture level at any farm?, Groundtruth answer: Second month, Predicted answer: 2.0, Output: True
	Example of 5% error tolerance:
Task prompt	Question: What was the highest closing price during the observed period?, Groundtruth answer: 147.10, Predicted answer: 148.00, Output: True
	Example of 5% error tolerance:
	Question: By how much does the percentage of viewers for Drama exceed that for Documentary/Educational?, Groundtruth answer: 20, Predicted answer: 40%, Output: False
	Imagine you are an intelligent teacher. Thoroughly read both the reference answer and the predicted answer to ensure a clear understanding of the information provided. Assess the accuracy of the predictions, noting that a prediction will be considered correct if it conveys the same meaning or value as the reference answer. Please ignore %, \$, and other units. Note that if the ground truth answer is a numeric value, with or without a unit, apply a 5% error tolerance to the answer. Your response should be either "True" or "False".
	\n\nQuestion: {q}\n\nGroundtruth answer: {g}\n\nPredicted answer: {p}.

Figure A7: Input prompt example for GPT-Acc. q, g, and p in the task prompt denote the question, ground truth answer, and predicted answer, respectively.

Table A8: **Comprehensive evaluation on our benchmark.** Note that R-Acc and GPT-Acc denote relaxed accuracy and GPT accuracy, respectively.

Method	Literal QAs		Inferential QAs		Reasoning QAs		Overall		Extraction	
	R-Acc	GPT-Acc	R-Acc	GPT-Acc	R-Acc	GPT-Acc	R-Acc	GPT-Acc	F1	RNSS
GPT-4o	43.98	47.62	57.23	59.43	23.08	26.15	41.40	44.40	66.10	88.56
LLaVA _{13B}	8.96	8.68	24.91	21.76	2.46	2.46	12.11	10.97	9.22	45.82
ChartLlama _{13B}	21.29	21.01	38.05	35.22	8.92	8.62	22.60	21.50	11.40	64.14
ChartInstruct _{7B}	26.33	21.07	43.40	32.81	13.23	<u>25.23</u>	27.5	26.15	8.65	39.73
ChartAst _{13B}	27.64	24.09	38.22	32.70	14.65	17.97	26.55	24.87	16.73	68.04
CHOPINLLM	<u>34.45</u>	<u>44.82</u>	<u>56.92</u>	<u>58.18</u>	<u>21.85</u>	21.23	<u>37.50</u>	<u>41.40</u>	<u>28.42</u>	<u>75.44</u>

workers with a chart image along with the corresponding question and answers. Human workers are asked to filter the QA pairs based on two criteria: (1) Answerability: whether the question is answerable given the chart image, and (2) Correctness: whether the provided answer is correct. After collecting the feedback, we perform final data filtering to obtain the benchmark set.

C.4 EVALUATION METRICS

To quantitatively analyze the performance on our benchmark, we adopt the relaxed accuracy metric for numeric answers, allowing a 5% margin of error from the exact value, and use exact match for non-numeric answers as per the standard in previous studies. Since the output of the MLLM model can be open form, following previous works (Liu et al., 2024a; Han et al., 2023; Xia et al., 2024), we also provide GPT-accuracy (GPT-Acc) using GPT model to further verify the performance. In Fig. A7, we show how we leverage the LLM to measure the accuracy by providing the detail prompts.⁶ Lastly, for underlying data understanding task, we follow previous work Deplot (Liu et al., 2022a) and measure performance using F1 score of Relative Mapping Similarity (RMS) and Relative Number Set Similarity (RNSS) to evaluate numeric accuracy and raw data similarity, respectively.

C.5 PERFORMANCE OF EXISTING MODELS

We evaluate existing models on our benchmark, and the results are provided in Table A8. We compare our model with four previous works, including ChartLlama Han et al. (2023), LLaVA (Liu

⁶GPT-4o-mini (2024-07-18)

Table A9: Performance comparison with synthetic data using in different training stages.

Idx	Model	Using synthetic data in	ChartQA	
			human	augmented
(1)	LLaVA-7B	-	36.00	67.44
(2)	LLaVA-7B	stage 3	37.60	70.40
(3)	LLaVA-7B	stage 1 & 2	52.28	87.68

et al., 2024b), ChartInstruct (Masry et al., 2024a), and ChartAst (Meng et al., 2024). As shown in the table, our model consistently outperforms all existing works across three different QA levels and the raw data extraction task. We note that, compared to existing benchmarks, our benchmark includes chart images with a greater variety of chart types (i.e., 20 different chart types) and features comprehensive QAs for each image, along with raw data extraction to assess the fundamental understanding of scientific charts. Therefore, outperforming previous works demonstrates that our model has broader understanding of the chart domain and a more comprehensive understanding of the underlying data. Lastly, we also evaluate the GPT model on our benchmark. We find that while the GPT model excels at capturing global concepts (i.e., inferential QAs), its performance on raw data understanding tasks (i.e., literal QAs and Extraction) is below 50% accuracy. Moreover, for reasoning QAs, it achieves only $\approx 25\%$ accuracy, highlighting its lack of mathematical reasoning ability with chart data. We note that evaluating GPT-4o vision on the entire benchmark would be expensive. To deal with this while having fair comparison, all results in this table are evaluated on the same small subset of the benchmark, consisting of 1,000 randomly sampled chart QA pairs.

D EXPERIMENTAL RESULTS

D.1 EFFECTIVENESS OF SYNTHETIC DATA IN DIFFERENT TRAINING STAGES

In the experiments of main paper, we showcase that chart data generated by text-only LLMs enhances MLLM learning in chart understanding. Here, we compare the performance of using the same data amount in the third stage of LoRA downstream fine-tuning, similar to Chartllama Han et al. (2023). The results are in Table A9. Specifically, Model (1) is the baseline, trained with LLaVA data in pre-training and fine-tuning stages, followed by LoRA fine-tuning on ChartQA. Model (2) is similar but includes synthetic data with ChartQA in the LoRA fine-tuning stage. Model (3), our best 7B model, uses the same synthetic data as Model (2) but incorporates it in pretraining and fine-tuning before LoRA fine-tuning on ChartQA. As shown in Table A9, Model (2) shows an $\approx 2\%$ improvement over Model (1), indicating the benefit of synthetic data. However, Model (3) shows a significant improvement over Model (2), suggesting synthetic data is more effective in fundamental training rather than fine-tuning. We propose two possible reasons for this. First, alignment issues cannot be effectively resolved via LoRA tuning, as it only adjusts a small portion of the model’s parameters. Second, the output preference of synthetic data may differ from that of the downstream dataset. Joint tuning might shift the output preference away from the downstream task, resulting in limited performance improvement.

D.2 SCALING LAW EXPERIMENTS

When fine-tuning models using generated data, a common question arises: How much data or how many training tokens should be used to achieve optimal performance for a given model? This question is similar to the one often asked in LLM research regarding training compute-optimal models (Hoffmann et al., 2022; Kaplan et al., 2020). Following previous works (Dubey et al., 2024; Hoffmann et al., 2022) on model scaling studies, our goal is to explore this by estimating the relationship between computational cost, model size, and the number of training tokens, providing guidance for future research on model and data selection within a compute budget. We assume a power-law relationship between computation and model size, as described in previous works (Clark et al., 2022; Kaplan et al., 2020). Specifically, we approach this problem from two angles: (1) FLOPs vs. parameters, where we fix the model size and vary the number of training tokens, and (2) FLOPs vs. training

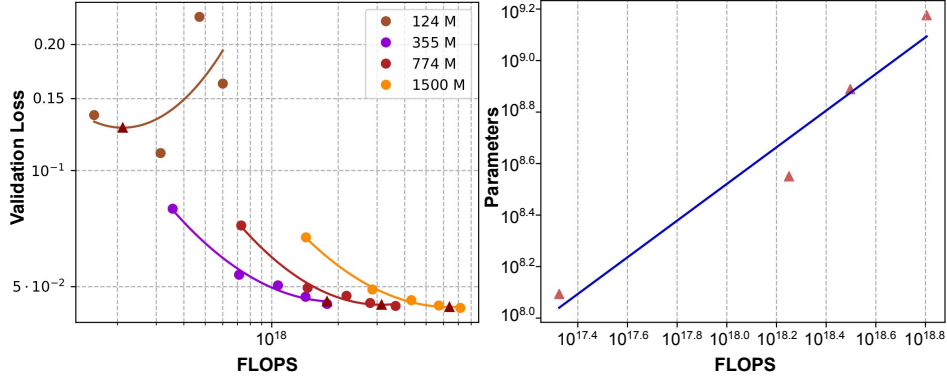


Figure A8: Training curve envelope showing the relationship between FLOPs and parameters, with model size fixed and training tokens varied.

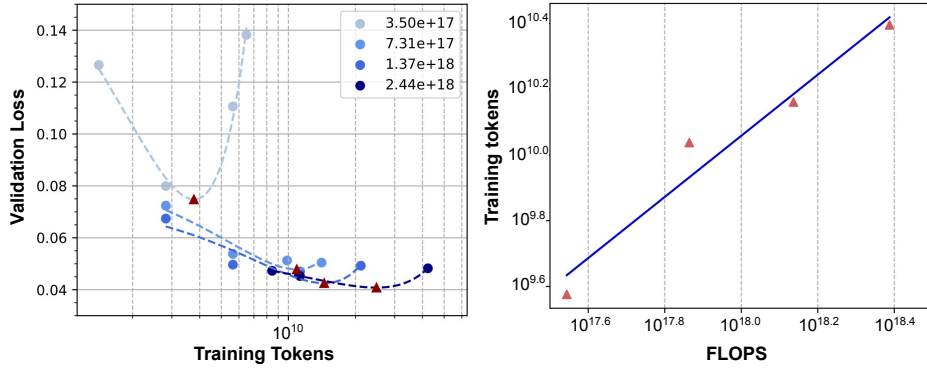


Figure A9: Training curve envelope showing the relationship between FLOPs and training tokens, with FLOP count fixed and model size varied.

tokens, where we vary the model size while keeping the training FLOP count constant. These two approaches are discussed in detail in the following sections.

FLOPs vs. Parameters: Fixing Model Size and Varying Training Tokens To determine the relationship between parameters and FLOPs for compute-optimal models, we need to identify the FLOPs required to reach the optimal point on the validation set. Specifically, we train models with varying numbers of tokens, find the optimal point (estimated lowest loss), and measure its corresponding FLOPs. With these optimal loss points and their respective training FLOPs, we can leverage the power-law relationship to deduce the relationship between parameters and FLOPs. The results are shown in Fig. A8. In practice, to construct smaller multimodal LLMs for testing, we created four different MLLMs by combining the CLIP vision tower with four GPT-2 models of varying sizes. Similar to LLaVA, a projector is employed to connect the vision and textual modalities. We then trained each of these models with five different token counts on our generated datasets, enabling us to collect five distinct points. Using a second-order polynomial fit, we estimate the optimal points. As shown in Fig. A8, once we have the optimal points for each model and the corresponding FLOPs, we can plot FLOPs against parameter size on a logarithmic scale. Using these interpolants, we derive a mapping from any given FLOP count C to the most efficient model size N . Applying the power-law, we obtain the relationship $N_{opt} = C^\alpha$, where $\alpha \approx 0.72$.

FLOPs vs. Training Tokens: Varying Model Size and Keeping Training FLOP Count Constant Similarly, to determine the relationship between training tokens and FLOPs for compute-optimal models, we need to identify the FLOPs for the optimal point on the validation set. Unlike the previous approach, here we keep the training FLOP count constant and vary the model size. Specifically, we choose four different FLOP counts and train the four previously mentioned models until they reach the respective FLOP thresholds. We then plot the validation loss against the number of training tokens for each FLOP count, as shown on the left side of Fig. A9. As with FLOPs vs. parameters, we

identify the optimal point and its corresponding training tokens, and we can then plot FLOPs against training tokens on a logarithmic scale. Using these interpolations, we establish a mapping from any given FLOP count C to the most efficient number of training tokens D . Applying the power-law, we derive the relationship $D_{opt} = C^\alpha$, where $\alpha \approx 0.9293$. Extrapolating this scaling law to 6.188e19 FLOPs suggests training a 13B parameter model on 5.29e12 tokens, which is approximately 7 times the amount of data used to train a 1.5B model.

E SOCIAL IMPACT

Our model is capable of chart understanding and can interpret the raw data of a chart like a human, without relying on annotations, while also performing various levels of QA tasks. Thus, our model can be used in many data analysis scenarios, such as market research, healthcare trend analysis, and other data science areas. With the help of our model, humans can process large volumes of chart data more efficiently, make informed decisions, and enhance reporting accuracy. While our model provides benefits in chart understanding and analysis, there are potential negative impacts. For instance, it could be employed to create misleading data visualizations or generate false narratives when combined with other LLM tools. These fake charts and pieces of information can negatively affect decision-making processes.

F LIMITATION

In this paper, we propose an MLLM model for chart understanding, fundamentally trained on synthetic data. However, since the synthetic data generated by LLMs cannot be perfect, sometimes incorrect data can be introduced into the dataset and may not be filtered out by our filtering process. These data can result in misalignments and incorrect mappings during pre-training and fine-tuning, potentially leading to incorrect responses and hallucinations. Thus, the performance of our chart MLLMs is limited by the LLMs' generation capabilities. We can potentially include more advanced LLMs in the data generation pipeline to reduce the occurrence of incorrect data. Moreover, another limitation of our model is that it currently supports understanding only 18 chart types. However, there are many more chart types in the real world. Developing an open-domain, versatile chart understanding MLLM remains a task for future work.

G EXAMPLES FOR JSON TEMPLATE AND READMEs

G.1 EXAMPLE 1

Template JSON

```
{ "chart_title": "Chart Title", "x_axis_label": "X-Axis Label", "y_axis_label": "Y-Axis Label", "data": [
  { "category": "Category 1", "value": 0 }, { "category": "Category 2", "value": 0 }, { "category":
    "Category 3", "value": 0 } ] }
```

README

The JSON template provided below is designed to generate data for a 'Bar Chart' chart. This template includes all the necessary attributes and labels to ensure consistent data formatting for the chart.

- 'chart_title': This attribute represents the title or name of the bar chart. Please replace the placeholder text with an appropriate title for your chart.

- 'x_axis_label': This attribute represents the label for the x-axis of the chart. Please replace the placeholder text with an appropriate label.

- 'y_axis_label': This attribute represents the label for the y-axis of the chart. Please replace the placeholder text with an appropriate label.

- 'data': This attribute represents the data points for the chart. It is an array of objects, where each object represents a category or group and its corresponding value.

For a simple example, the template can be filled as follows:

Definition of a 'Bar Chart' chart:

A bar chart, also known as a bar graph, is a visualization tool that uses rectangular bars to represent data. Each bar represents a category or group, and the length or height of the bar corresponds to the value it represents. Bar charts are commonly used to compare categorical data or show the distribution of data across different categories.

G.2 EXAMPLE 2

Template JSON

```
{ "chart": { "title": "Chart Title", "xAxisLabel": "X-axis Label", "yAxisLabel1": "Y-axis Label 1",
  "yAxisLabel2": "Y-axis Label 2", "datasets": [ { "name": "Dataset 1", "type": "line", "data": [] }, {
    "name": "Dataset 2", "type": "bar", "data": [] } ] } }
```

README

This JSON file template is designed for generating datasets for a 'Multi-axes Line Bar Chart' chart. It includes the following attributes:

1. chart:

- title: (string) The title of the chart.

- xAxisLabel: (string) The label for the X-axis.

- yAxisLabel1: (string) The label for the Y-axis corresponding to the line chart.

- yAxisLabel2: (string) The label for the Y-axis corresponding to the bar chart.

2. datasets:

- name: (string) The name or label of the dataset.

- type: (string) The type of chart component for the dataset. Can be 'line' or 'bar'.

- data: (array) The array to store the data points for the dataset. Placeholder data should be added here.

The template provides a structure to accommodate both simple and complex examples of a Multi-axes Line Bar Chart. Additional datasets can be added within the "datasets" array.

Please ensure that the generated data adheres to the structure of this JSON template, including the attribute names and data types, to ensure consistency when plotting the chart using Python.

Definition of a 'Multi-axes Line Bar Chart':

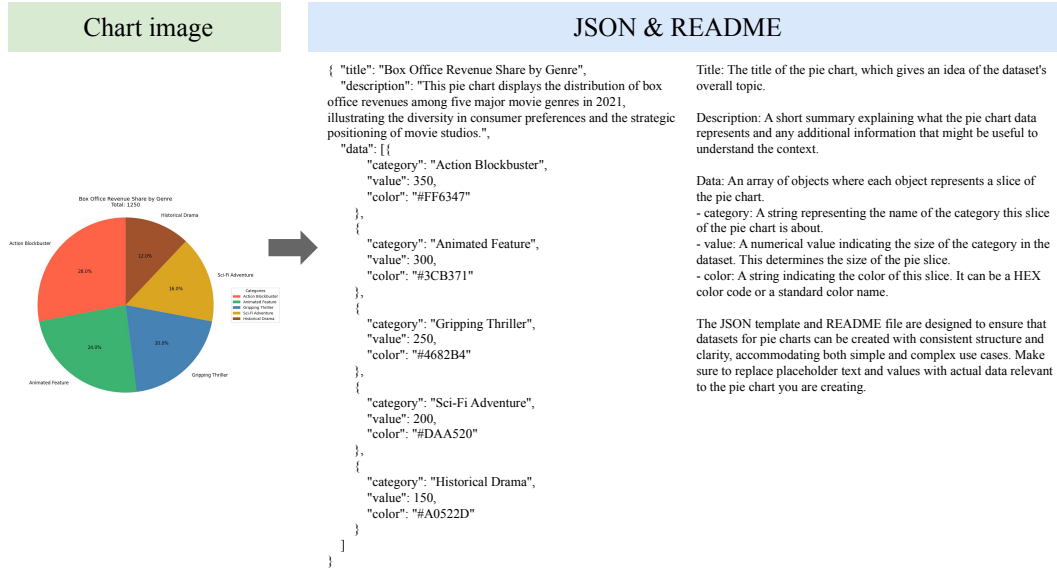
A multi-axes line bar chart is a type of chart that combines both line and bar charts in a single visualization. It allows for the comparison of multiple datasets that have different scales or units of measurement. This chart type uses multiple y-axes, one for each dataset, to display the corresponding line and bar components.

[illegible]

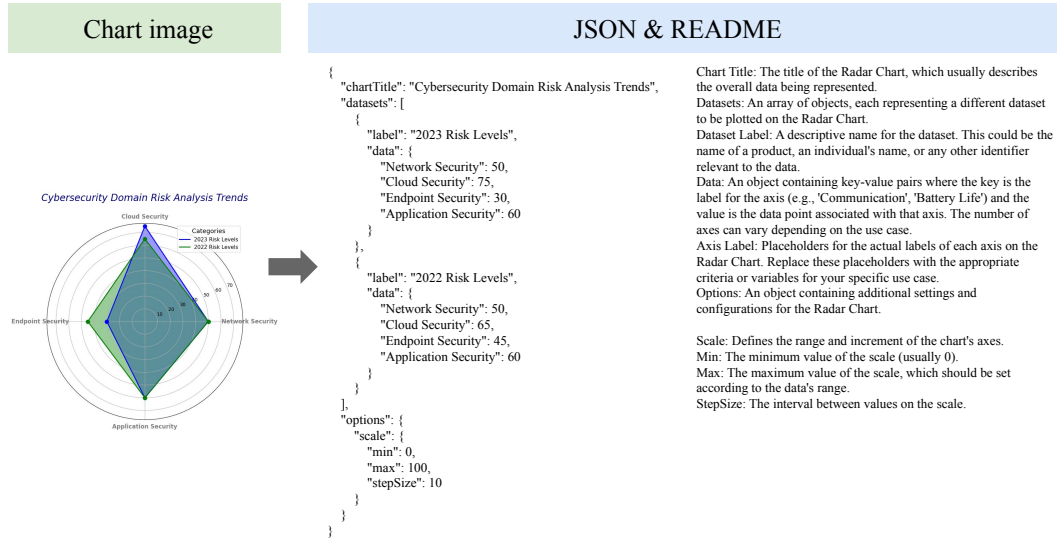
11

I EXAMPLES OF AUGMENTED QAS

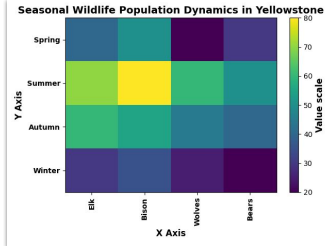
I.1 JSON-ONLY QA: EXAMPLE 1



I.2 JSON-ONLY QA: EXAMPLE 2



I.3 DATA-DRIVEN QA: EXAMPLE 1



An example of data extraction QAs

Can you please extract the raw data of this chart?

```
{ "title": "Seasonal Wildlife Population Dynamics in Yellowstone", "xAxis": { "title": "Species", "categories": [ "Elk", "Bison", "Wolves", "Bears" ] }, "yAxis": { "title": "Seasons", "categories": [ "Spring", "Summer", "Autumn", "Winter" ] }, "colorScale": { "minValue": 0, "maxValue": 100, "startColor": "#FFFFFF", "endColor": "#FF6347" }, "data": [ { "x": 0, "y": 0, "value": 40 }, { "x": 1, "y": 0, "value": 50 }, { "x": 2, "y": 0, "value": 60 }, { "x": 3, "y": 0, "value": 55 }, { "x": 0, "y": 1, "value": 70 }, { "x": 1, "y": 1, "value": 80 }, { "x": 2, "y": 1, "value": 60 }, { "x": 3, "y": 1, "value": 50 }, { "x": 0, "y": 2, "value": 50 }, { "x": 1, "y": 2, "value": 60 }, { "x": 2, "y": 2, "value": 60 }, { "x": 3, "y": 2, "value": 50 }, { "x": 0, "y": 3, "value": 30 }, { "x": 1, "y": 3, "value": 40 }, { "x": 2, "y": 3, "value": 30 }, { "x": 3, "y": 3, "value": 20 } ] }
```

Which season generally shows the highest population densities across all species?

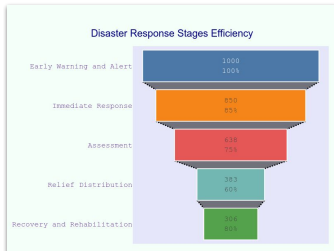
Long answer

Reviewing the heatmap data, Summer has high values for all species: Elk (70), Bison (80), Wolves (60), and Bears (50). Thus, Summer generally shows the highest population densities.

Short answer

Summer

I.4 DATA-DRIVEN QA: EXAMPLE 2



An example of data extraction QAs

Can you please extract the raw data of this chart?

```
{ "chart": { "title": "Disaster Response Stages Efficiency", "description": "This funnel chart maps the stages of disaster response from the initial alert to recovery, highlighting the number of actions completed and the conversion rates between each stage. The cyclical trend observed suggests repeated challenges in transitioning efficiently from one stage to the next." }, "stages": [ { "stage": "Early Warning and Alert", "value": 1000, "conversionRate": "85%" }, { "stage": "Immediate Response", "value": 850, "conversionRate": "75%" }, { "stage": "Assessment", "value": 638, "conversionRate": "60%" }, { "stage": "Relief Distribution", "value": 383, "conversionRate": "80%" }, { "stage": "Recovery and Rehabilitation", "value": 306, "conversionRate": null } ] }
```

What is the sum of conversion rates for all stages where it is mentioned?

Long answer

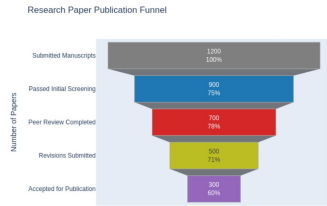
The sum of conversion rates for all stages mentioned is calculated by adding the rates: 85% (Early Warning and Alert) + 75% (Immediate Response) + 60% (Assessment) + 80% (Relief Distribution) = 300%.

Short answer

300%

J EXAMPLES OF DATA PROMPTING

J.1 EXAMPLE 1



Without data extraction QA in the inference

What is the acceptance rate of manuscripts that have been revised?

50%

With data extraction QA in the inference

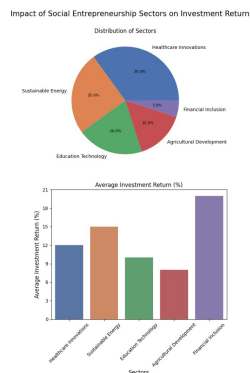
Retrieve the raw data from the image and organize it into a JSON dictionary.

```
{chart: {'title': 'Research Paper Publication Funnel', 'xAxisLabel': 'Stage', 'yAxisLabel': 'Number of Papers', 'data': [{'stage': 'Submitted Manuscripts', 'count': 1200}, {'stage': 'Passed Initial Screening', 'count': 900}, {'stage': 'Peer Review Completed', 'count': 700}, {'stage': 'Revisions Submitted', 'count': 500}, {'stage': 'Accepted for Publication', 'count': 300}]}}
```

What is the acceptance rate of manuscripts that have been revised?

60%

J.2 EXAMPLE 2



Without data extraction QA in the inference

What proportion of the total distribution is made up by 'Sustainable Energy' and 'Healthcare Innovations' combined?

45%

With data extraction QA in the inference

Retrieve the raw data from the image and organize it into a JSON dictionary.

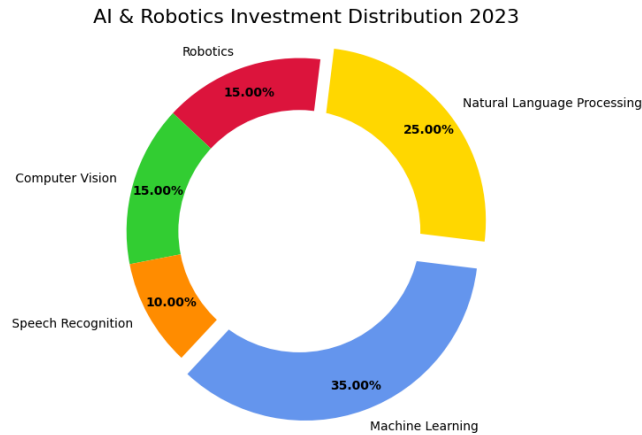
```
{chart_title: 'Impact of Social Entrepreneurship Sectors on Investment Returns', 'x_axis_label': 'Sectors', 'y_axis_label_pie': 'Distribution of Sectors', 'y_axis_label_bar': 'Average Investment Return (%)', 'data': [{'category': 'Healthcare Innovation', 'value_pie': 35, 'value_bar': 12}, {'category': 'Sustainable Energy', 'value_pie': 5, 'value_bar': 15}, {'category': 'Education Technology', 'value_pie': 20, 'value_bar': 10}, {'category': 'Agricultural Development', 'value_pie': 15, 'value_bar': 8}, {'category': 'Financial Inclusion', 'value_pie': 5, 'value_bar': 20}]}
```

What proportion of the total distribution is made up by 'Sustainable Energy' and 'Healthcare Innovations' combined?

60%

K EXAMPLES FROM OUR BENCHMARK

K.1 EXAMPLE 1



JSON Data

```
{'chart': {'type': 'donut', 'title': 'AI & Robotics Investment Distribution 2023'}, 'data': {'labels': ['Machine Learning', 'Natural Language Processing', 'Robotics', 'Computer Vision', 'Speech Recognition'], 'datasets': [{'label': 'Investment Proportions', 'data': [35, 25, 15, 15, 10], 'backgroundColor': ['#6495ED', '#FFD700', '#DC143C', '#32CD32', '#FF8C00']}]}}
```

Literal Question

Question: How much less investment did Speech Recognition receive compared to Natural Language Processing?

Long Answer: Speech Recognition received 15% less investment than Natural Language Processing, with Speech Recognition at 10% and Natural Language Processing at 25%.

Short Answer: 15%

Inferential Question

Question: What two sectors together make up half of the total investment?

Long Answer: Machine Learning and Natural Language Processing together make up half of the total investment, with percentages of 35% and 25% respectively.

Short Answer: Machine Learning and Natural Language Processing

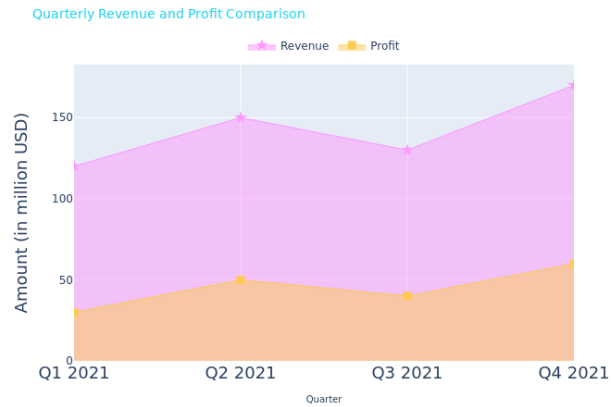
Reasoning Question

Question: What fraction of the total investment is allocated to fields other than Machine Learning?

Long Answer: Fields other than Machine Learning receive a combined total of 65% of the investment, which is equivalent to the fraction 65/100 or 13/20 of the total investment.

Short Answer: 13/20

K.2 EXAMPLE 2



JSON Data

```
{'chart': {'title': 'Quarterly Revenue and Profit Comparison', 'xAxisLabel': 'Quarter', 'yAxisLabel': 'Amount (in million USD)'}, 'data': [{'category': 'Revenue', 'values': [{'x': 'Q1 2021', 'y': 120}, {'x': 'Q2 2021', 'y': 150}, {'x': 'Q3 2021', 'y': 130}, {'x': 'Q4 2021', 'y': 170}]}, {'category': 'Profit', 'values': [{'x': 'Q1 2021', 'y': 30}, {'x': 'Q2 2021', 'y': 50}, {'x': 'Q3 2021', 'y': 40}, {'x': 'Q4 2021', 'y': 60}]}]}
```

Literal Question

Question: What was the Revenue in Q3 2021?

Long Answer: In Q3 2021, the Revenue was 130 million USD as shown on the chart.

Short Answer: 130 million USD

Inferential Question

Question: Which quarter had the highest ratio of Profit to Revenue?

Long Answer: To determine the highest ratio of Profit to Revenue, we compare the ratios for each quarter. The highest ratio is in Q2 2021, with Profit at 50 million USD and Revenue at 150 million USD, giving a ratio of 1:3.

Short Answer: Q2 2021

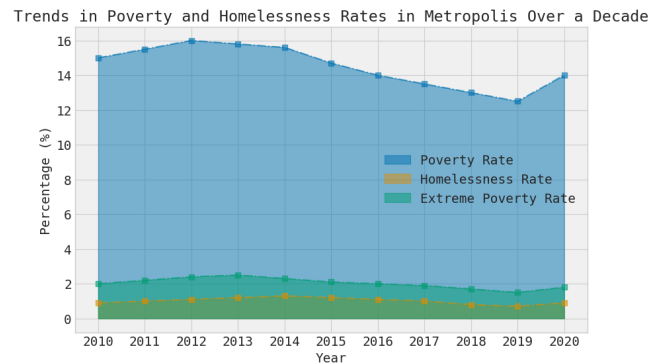
Reasoning Question

Question: If Profit in Q1 2022 is expected to be 20% higher than Q4 2021, what would be the expected Profit?

Long Answer: If Profit in Q1 2022 is expected to be 20% higher than Q4 2021's 60 million USD, the expected Profit would be 60×1.20 , which is 72 million USD.

Short Answer: 72 million USD

K.3 EXAMPLE 3



JSON Data

```
{'chart': {'title': 'Trends in Poverty and Homelessness Rates in Metropolis Over a Decade', 'xAxisLabel': 'Year', 'yAxisLabel': 'Percentage (%)'}, 'data': [{'category': 'Poverty Rate', 'values': [{'x': '2010', 'y': 15.0}, {'x': '2011', 'y': 15.5}, {'x': '2012', 'y': 16.0}, {'x': '2013', 'y': 15.8}, {'x': '2014', 'y': 15.6}, {'x': '2015', 'y': 14.7}, {'x': '2016', 'y': 14.0}, {'x': '2017', 'y': 13.5}, {'x': '2018', 'y': 13.0}, {'x': '2019', 'y': 12.5}, {'x': '2020', 'y': 14.0}]}, {'category': 'Homelessness Rate', 'values': [{'x': '2010', 'y': 0.9}, {'x': '2011', 'y': 1.0}, {'x': '2012', 'y': 1.1}, {'x': '2013', 'y': 1.2}, {'x': '2014', 'y': 1.3}, {'x': '2015', 'y': 1.2}, {'x': '2016', 'y': 1.1}, {'x': '2017', 'y': 1.0}, {'x': '2018', 'y': 0.8}, {'x': '2019', 'y': 0.7}, {'x': '2020', 'y': 0.9}]}, {'category': 'Extreme Poverty Rate', 'values': [{'x': '2010', 'y': 2.0}, {'x': '2011', 'y': 2.2}, {'x': '2012', 'y': 2.4}, {'x': '2013', 'y': 2.5}, {'x': '2014', 'y': 2.3}, {'x': '2015', 'y': 2.1}, {'x': '2016', 'y': 2.0}, {'x': '2017', 'y': 1.9}, {'x': '2018', 'y': 1.7}, {'x': '2019', 'y': 1.5}, {'x': '2020', 'y': 1.8}]}]}
```

Literal Question

Question: In which year did the Poverty Rate reach its lowest value?

Long Answer: According to the chart data, the Poverty Rate reached its lowest value in 2019 at 12.5%.

Short Answer: 2019

Inferential Question

Question: Did any category show a consistent decline over the entire decade?

Long Answer: No category showed a consistent decline over the entire decade. While Poverty Rate and Extreme Poverty Rate generally declined until 2019, they both increased in 2020, and the Homelessness Rate fluctuated throughout the decade.

Short Answer: No

Reasoning Question

Question: What is the average annual decrease in the Poverty Rate from 2010 to 2019?

Long Answer: From 2010 to 2019, the Poverty Rate decreased from 15.0% to 12.5%. This is a total decrease of 2.5 percentage points over 9 years, which gives an average annual decrease of about 0.278 percentage points per year.

Short Answer: Approximately 0.278 percentage points per year