Synthesize Step-by-Step: Tools, Templates and LLMs as Data Generators for Reasoning-Based Chart VQA

Zhuowan Li[1] * Bhavan Jasani[2] * Peng Tang[2] Shabnam Ghadar[2]
[1] Johns Hopkins University [2] AWS AI Labs
zli110@jhu.edu {bjasani, tangpen, shabnam}@amazon.com

Abstract

Understanding data visualizations like charts and plots requires reasoning about both visual elements and numerics. Although strong in extractive questions, current chart visual question answering (chart VQA) models suffer on complex reasoning questions. In this work, we address the lack of reasoning ability by data augmentation. We leverage Large Language Models (LLMs), which have shown to have strong reasoning ability, as an automatic data annotator that generates question-answer annotations for chart images. The key innovation in our method lies in the Synthesize Step-by-Step strategy: our LLM-based data generator learns to decompose the complex question into step-by-step sub-questions (rationales), which are then used to derive the final answer using external tools, i.e. *Python. This step-wise generation procedure is trained on synthetic data generated using a template-based QA generation pipeline. Experimental results highlight the significance of the proposed step-by-step generation. By training with the LLM-augmented data (*LAMENDA*), we significantly enhance the chart VQA models, achieving the state-of-the-art accuracy on the ChartQA and PlotQA datasets. In particular, our approach improves the accuracy of the previous state-of-the-art approach from 38% to 54% on the human-written questions in the ChartQA dataset, which needs strong reasoning. We hope our work underscores the potential of synthetic data and encourages further exploration of data augmentation using LLMs for reasoning-heavy tasks.*

1. Introduction

Data visualizations like charts and plots play an important role in real-world data analysis applications. Unlike natural images, chart are text-heavy and data-driven, thus requiring better visual perception (*e.g.* OCR) and cognitive reasoning (*e.g.* math calculation, matching the legends with bars). For example, to answer "what is the total of Democrats and Republicans in 2010" in Fig. 1, the model needs to recognize the texts, match the legends of "democrats" and "republicans" to the bars, then calculate the sum. This is a challenging task requiring multimodal perception and reasoning.

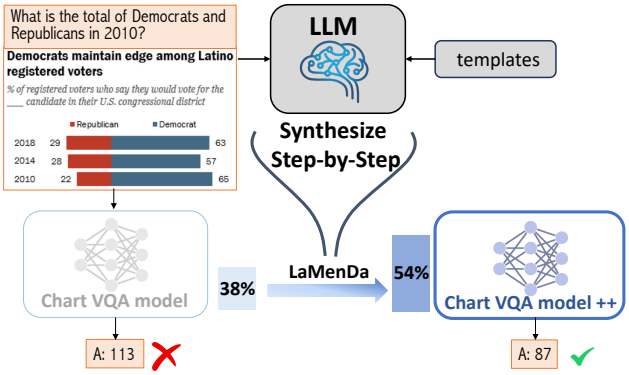


Figure 1. Existing chart VQA models struggle with complex reasoning questions. We attribute this to limited reasoning questions in existing datasets and address it by data augmentation. We finetune an LLM-based data generator that automatically generates question-answer annotations given a chart image. Our key innovation is Synthesize Step-by-Step, which breaks complex questions down into easy steps that could be solved using external tools. We use templates to train the LLM. Training with LLM-augmented data, LAMENDA, greatly enhances the chart VQA models.

Compounding the complexity, the inherent difficulty of chart reasoning also amplifies the cost of collecting human annotations. Most existing datasets are synthetically generated using simple rules and heuristics, posing a risk of model overfitting. Among the datasets, only ChartQA [40] contains complex human-annotated question-answer pairs, although there are very few of those per image. For example, for the image in Fig. 1, while a lot more questions can be asked, only two questions are annotated (What is the total of Republicans and Democrats in 2010? Which color represents Republican?). Importantly, these human-written questions pose great challenges for contemporary models. For example, the state-of-the-art model [35] achieves only 38% accuracy when confronted with human-written ques-

*indicates equal contributions. The work was done when Zhuowan Li was an intern at Amazon.

tions that require multi-step complex reasoning. Approximately half of the errors, as reported in [35], are due to reasoning failures.

In this work, we propose to leverage large language models (LLMs) as automatic data annotators for chart reasoning. Recently, LLMs have demonstrated strong reasoning ability for complex tasks [27, 58, 59, 63, 68, 69]. Our motivation is to take advantage of this strong reasoning ability and generate question-answer pairs using LLMs. This LLM-generated data can be used to train the chart VQA models, thus improving their reasoning ability.

The key innovation in our method lies in the *Synthesize Step-by-Step* strategy, which leverages LLMs to generate data in a step-by-step manner. Similar to human annotators who are trained to break down complex tasks into easier subtasks, our LLM-based data generator generates step-by-step. Instead of generating question-answer all at once, the generator learns to generate the question first, then decompose the complex question into step-by-step sub-questions (rationales) and answer them one by one. Finally, the answers of the sub-questions are combined to derive the final answer using external tools like calculators or Python code. Specifically, we feed the image features extracted with a ViT [12] backbone into an LLM using a simple trainable projection layer. To supervise the training of this projection layer, we construct a template-based training corpus generated using a template-based QA generation pipeline. Importantly, as we will show in the paper, the step-by-step synthesizing, trained using these template-based rationales, is critical for generating good-quality data.

Experiments showcase the effectiveness of Synthesize Step-by-Step. Our LLM-augmented Data, LaMenDa, is used to train the downstream chart VQA models and its effectiveness is shown on two challenging datasets, ChartQA and PlotQA with the state-of-the-art accuracy. Notably, we significantly improve the performance from 38% to 54% on the challenging human-written questions in ChartQA dataset, which requires complex reasoning. We also demonstrate the advantage of step-wise generation over straightforward generation. With the results, we hope our work underscores the potential of synthetic data and encourages further exploration of data augmentation using LLMs for reasoning-heavy tasks.

2. Related Work

Multimodal models for text-heavy images fall into two categories: OCR-dependent or OCR-free. Models like LayoutLM [61], PaLI(-X) [7, 8], ChartBERT [1], DocFormerv2 [4], MQMA [53], DEED [54] rely on external OCR systems like ChartOCR [38] as either model input or training objectives. Recent top performing models, like Donut [26], Dessurt [10], Pix2struct [28], shift away from OCR thanks to vision transformers [11]. Pix2Struct [28] is a transformer encoder-decoder model for multiple VQA tasks, pretrained with an HTML-masked prediction objective, using a large amount of web page screenshots. MATCHA [35] and DEPLOT [34] are two recent follow-up works, tuning Pix2struct with chart-specific tasks like chart de-rendering and math reasoning, which are used in our work.

Datasets for chart VQA. Earlier datasets like DVQA [22], FigureQA [23], LeafQA [5] are based on synthetically generated images and templatic questions, which are easy to solve and prior models achieve $> 95\%$ accuracy. PlotQA [41] is also a synthetic dataset but contains open-vocabulary questions. ChartQA [40] is a recent dataset containing real images with human-written questions, which is much more challenging. Our work is mainly based on ChartQA and PlotQA, following the settings in [34, 35].

LLMs for multimodal tasks. There is a surge of interest in extending LLMs into the vision-and-language domain. Recent works align the image features into the input space of the pretrained LLMs using a lightweight module, *e.g.* Frozen [56], Flamingo [2], LLaVA [36], Otter [29], MiniGPT-4 [70], BLIP-2 [30], AdaptorV2 [13], etc. For example, LLaVA [36] uses a simple projection layer to align the modalities and build a visually-finetuned multimodal model; LLaVAR [67] extends it onto text-heavy images.

Reasoning step-by-step has been studied for a long time in the multimodal field. Compositional models like neural modular networks [3, 15, 19] or neural symbolic reasoning [32, 33, 39, 64] decompose complex visual questions into reasoning steps, and compose modules to solve the steps. For LLMs, chain-of-thought prompting is a simple yet effective method to boost the reasoning performance [27, 58, 59, 63, 68, 69]. Recent works like VisProg [16] and ViperGPT [51] utilize LLMs as the planner to compose domain-specific models [37, 43, 62, 66] and external tools or APIs [25, 42, 45, 48, 50] for solving complex tasks.

LLMs for data generation has been recently studied for language tasks [9, 49, 65] or multimodal tasks [36, 70]. This relates to knowledge distillation [17], which teaches a smaller model by learning from a larger teacher model, benefiting semi-supervised learning [6, 57]. Distillation from LLMs [14, 18, 31, 44, 60] can be completed by either learning from the logits or from the generated data directly. Our work is related but different from these works in that we focus on step-by-step generation, combining LLMs with templates and tools, for the chart reasoning task.

3. Method

Given a chart plot, our goal is to build a data generator that generates complex reasoning questions with answers. The generated data can be used for training the downstream chart VQA models.

The overview of our LLM-based data generator is shown in Fig. 2. In this section, we describe the data generator in a

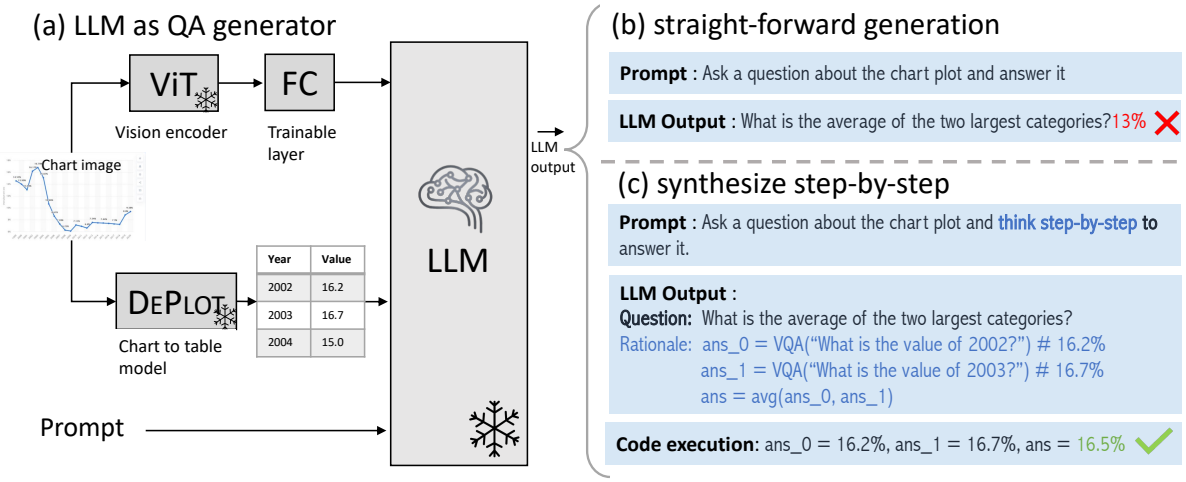


Figure 2. Method overview. (a) The architecture of the LLM-based question-answer generator. Image features, projected by a linear layer concatenated with the predicted data table and a prompt, are fed into LLM for QA generation. (b) Straight-forward generation, where questions with answers are generated in a straightforward way. (c) Synthesizing step-by-step, which breaks the question down into rationale programs and derives the answer by executing the program.

progressive manner. We first introduce an intuitive, straightforward method for data generation. Then, we introduce Synthesize Step-by-Step, which enhances the straightforward method with step-wise rationales. Finally, we introduce the settings for training the downstream chart VQA models using our generated QA data.

3.1. Architecture

The architecture of our LLM-based data generator is shown in Fig. 2 (a). We leverage MPT-7B [55] as the LLM for data generation. The inputs X to LLM is a concatenation of three components: the projected image features F, the underlying data table for the chart image T, and a natural language prompt P:

$$X = [F; T; P]$$

To extract the image features, we consider the pretrained visual transformer (ViT) [11] in CLIP [46]. The ViT takes in the image patches, and extracts features for each image patch. Then we use a trainable linear layer to project the extracted image features into the input embedding space of LLM. This architecture, *i.e.* linear layer for visual feature alignment, has been shown effective by recent multimodal large models like LLaVA [36].

The predicted data table is critical to encode the textual information in the image. Because the pretrained ViT (in CLIP [46]) is not specifically trained for text-heavy images, the image features have limited OCR ability. Therefore, we use DEPLOT [34], which is designed for information extraction from charts, to predict the underlying data table for the given chart image. The linearized data table, which is a sequence of word tokens, is fed into the LLM.

Finally, the projected image features, the data table, and a prompt like "Ask a question about the given chart plot and answer it" (rephrased in several different ways), are provided as the input to LLM, for question generation.

3.2. Straight-forward data generation

Given the above inputs, a straightforward manner for data generation is to have the LLM generate question and answer directly, as shown in Fig. 2 (b). The output Y is the generated question Q, followed by the corresponding answer Q:

$$Y = [Q; A]$$

This straightforward data generator can be trained using the existing chart VQA datasets, which contains chart images paired with question and answers. The model is trained with a next-token prediction loss. Note that the ViT and LLM are pretrained and kept frozen during training. The only trainable parameters are the linear projection layer[1].

As our experiments will show, this straightforward generation method yields reasonable results, generating questions and answers that are helpful for training the chart VQA models. However, the generated answers are noisy, which is improved using step-by-step generation, described next.

3.3. Synthesize step-by-step

Here we introduce *Synthesize Step-by-Step*, which shares the same model architecture as straight-forward generation, but performs data synthesizing in a step-by-step manner.

Instead of having the model generate answers directly, we train the LLM to generate them in a step-by-step manner. The LLM generates the question Q with the corresponding

[1]We also tried a two-stage training procedure, *i.e.* finetune the whole model after training the projection layer, but did not see performance gain.

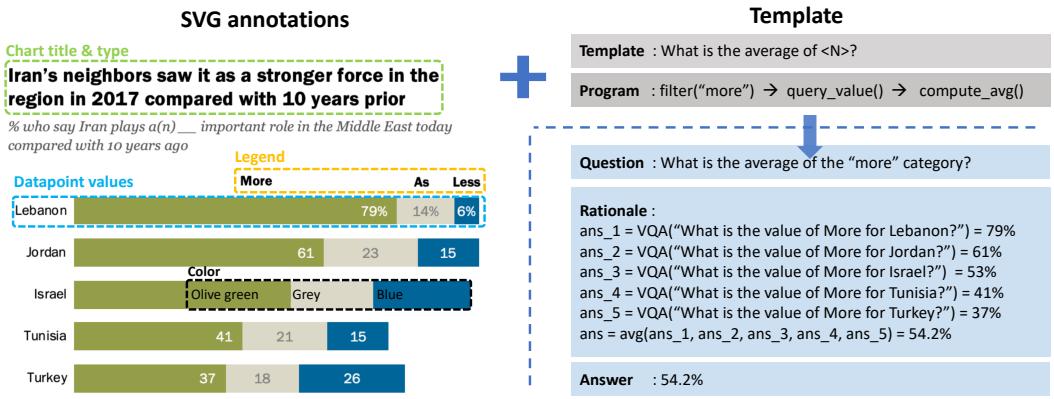


Figure 3. Example of template-based question generation. Given a manually created `template` with defined `program`, by querying the image `SVG annotations`, templatic `questions` and `answers` can be generated, with `rationales`.

rationales R. Then the answer A is derived by executing the rationales using python scripts:

$$Y = [Q; R], A = \text{Execute}(R)$$

As the example shows in Fig. 2 (c), the LLM outputs the generated question and a decomposition of this complex reasoning question into a series of atom reasoning steps, *i.e.* rationales. The rationales R a step-by-step executable program, where each step in the program can be either an atom extractive VQA questions (*e.g.* `ans_0=VQA("What is the value of 2002?")`), or a Python-like program call for math calculation, counting, comparison, etc. (*e.g.* `ans=avg(ans_0, ans_1)`). The rationale program is then parsed and executed using Python in order to derive the final answer. During execution, there are two ways to answer the atom extractive VQA questions: we can either use an off-the-shelf VQA model, or directly predict the answer using the LLM data generator itself. We experiment with both options and find the latter option to be more effective, as well as more simple, without requiring additional VQA models.

Training with rationales. To train the step-by-step generator, data *with rationales* is required. This is different from the straightforward generation, which only requires question-answer pairs to train. Because existing datasets do not contain the rationale annotations for each question, we introduce a template-based QA generation pipeline, which will be described in Sec. 3.4, to generate synthetic questions with rationales. With the template-generated data as the training corpus, we tune the generator for step-by-step generation. An additional advantage of training with template-generated data is that, in inference time, the generator can be prompted with different prompts for controllable generation. The prompts may specify the format or the type of the questions, *e.g.* "the question should start with how many.", "the question should ask about the colors in the image", etc. A complete list of the prompts can be found in Appendix.

Post hoc filtering. We take an additional post hoc filtering step to filter out the noisy questions. We use the the decoding score (for the whole generated sequence including question and rationales) as an indicator and filter out the questions with decoding scores lower than a threshold. Better filtering strategies, like using LLMs for data filtering, can be explored, which we will leave as future work.

3.4. Tuning corpus: template-based QA generation

Here we describe how we use a template-based pipeline to generate questions with rationales for training the LLM-based generator. The method is motivated by the success of synthetic compositional questions in VQA datasets like CLEVR [21], Super-CLEVR [33] and GQA [20].

Fig. 3 illustrates the template-based question generation method. We first manually create question templates, then instantiate the templates into questions and answers by querying the rich SVG annotations of the chart image (as shown in the dotted boxes in the figure). For example, the template "What is the average of the $\langle N \rangle$?" can be instantiated into "what is the average of the more category?". The argument $\langle N \rangle$ is instantiated into "the more category" according to the SVG annotations, which includes the category names, values, and visual elements like colors and bounding boxes. Moreover, for each template, we define a reasoning program (in a domain-specific language) containing a series of reasoning operations (*e.g.* `select_color, query_value`) that specifies the reasoning process needed to answer the question. The program is then instantiated into a step-by-step rationale.

Note that there are limitations to this template-based QA generation pipeline. The pipeline relies on the rich annotations extracted automatically from the source SVG files, and thus cannot generalize to unlabeled images. Moreover, even though some existing datasets like ChartQA provide such rich annotations crawled from websources, they are inevitably noisy, containing missing or incorrect values. In contrast, the LLM generator is more generalizable, being

able to generate data for images in the wild. Therefore, in our work, these template-based questions are used as the training corpus of the LLM-based generator, which is more generalizable and scalable.

3.5. Evaluation of generated data

To measure the usefulness of the generated QA, we use the generated data to train the chart VQA models. The accuracy improvement of chart VQA models, after training with the generated QA, is taken as the indicator of the data quality. We select MATCHA [35], which is a transformer encoder-decoder model pretrained on large-scale text-heavy images, as our primary evaluation model, considering its modern architecture and state-of-the-art performance on standard datasets[2].

Summary. Our data generation procedure contains the following steps: first, we generate templatic QA corpus with rationales using the rich SVG annotations, which is then used to tune the LLM-based data generator; next, the LLM-based generator generates questions with step-by-step rationales; then we use Python to execute the rationales and derive the answers; finally, this LLM-augmented data, LAMENDA, is used to train the downstream VQA model MATCHA. As the experiments will show, the augmented data leads to significant performance improvements.

4. Experiment

4.1. Set up

Dataset. Following [34, 35], we run experiments on two chart VQA datasets: ChartQA and PlotQA [3]. ChartQA [40] contains 21k real-world chart images crawled from four web sources, with 9k human-annotated (Human split) and 23k machine-augmented (Augmented split) QA pairs. The augmented questions are mostly extractive questions, while the human-annotated questions are free-form and require complex reasoning. PlotQA [41] is a large synthetic dataset containing 224k chart images generated from real-world online data, with 37M synthetic questions and answers divided into two splits: V1 (8M) and V2 (29M). In addition, to show the generalizability of our LLM data generator, we generate QA for images from additional chart captioning datasets, which do not provide QAs or rich SVG annotations. We use images from three datasets: Chart-to-Text [24], VisText [52] and ChartSumm [47]. Chart-to-Text contains 35k images from "Statistica" and 9k images from "Pew"; VisText contains 9k synthetic charts of various styles; ChartSumm contains 84k images from "Statistica" and "Knoema". Note that we only use the images from the chart captioning datasets without using the captions.

Generated data. Tab. 1 shows the statistics of the generated QA. To construct the template-based tuning corpus, we use 28 templates to generate 357k question-answer pairs for the ChartQA training images. The template list and the template QA can be found in the Appendix. We train our LLM data generator using the template-based corpus, and generate QAs for images in ChartQA, PlotQA, and the three chart captioning datasets. For chartQA training images, we generate 403k QAs, which reduces to 326k after step-by-step execution (dropping ones that cannot be executed) and filtering based on the decoding scores using a threshold of -10. For PlotQA training images, we generate 3M QAs, which reduces to 1.7M after execution and filtering. For the 137k chart images in the chart captioning datasets, we generate 1.6M QAs.

	Dataset	Images	Generated	Filtered
Template	ChartQA	18,317	356,606	-
LAMENDA	ChartQA	18,317	402,974	326,160
	PlotQA	157,070	3,455,539	1,726,822
	ChartCap	137,281	2,987,718	1,635,364

Table 1. Statistics of generated data, before and after filtering.

Implementation details. Using the 357k template QA, we tune the data generator for 5 epochs with a batch size of 128. We use a cosine learning rate schedule with a max learning rate of 2e-3. The training takes about 12 hours using 8 A100 GPUs. The trainable projection layer is initialized with the first stage checkpoint of LLaVA [36]. We select the best checkpoint at 14k iterations as our best generator. For MATCHA training, we take the pretrained checkpoints (base size) from HuggingFace[4] and finetune the models with a combination of our generated questions and the original ChartQA or PlotQA training questions. The training setting follows [28, 35]. We use a cosine scheduler with 500 warm-up steps and a max learning rate of 0.0001. The batch size is 256. Without special description, the models are trained for 10k iterations for ChartQA (combining human and augmented splits) and 20k iterations for PlotQA because it is much larger (V1 and V2 are trained separately). We set the max number of image patches as 4096. The ChartQA training takes 3 days on 8 32GB V100 GPUs. Considering the training time, in some experiments, we reduce the image patch number to 1024 for quick experiments in 12 hours with the same setting.

Evaluation metrics. We suggest that both strict accuracy and relaxed accuracy should be reported, especially for the ChartQA dataset. While existing works [34, 35, 40, 41] evaluate answers are evaluated using relaxed accuracy, which allows 5% error for numerical answers, we find that this relaxed accuracy metric is problematic for ChartQA.

[2] While there are two-stage models of better performance, which extracts information from charts then prompts LLMs like Codex to answer questions, MATCHA is the best single-stage model.

[3] There are other chart VQA datasets like FigureQA, DVQA, etc., but since they are out-dated and models reach $> 95\%$ accuracy, we focus on more challenging settings.

[4] https://huggingface.co/google/matcha-base

		Accuracy			Relaxed Accuracy		
		avg	human	augment	avg	human	augment
0	baseline (MATCHA [35])	47.76	30.21	65.31	58.54	39.58	77.50
1	syn_a	48.80	28.65	68.96	59.43	38.65	80.21
2	syn_{at} w/ VQA answer	47.76	28.44	67.08	57.97	37.81	78.12
3	syn_{at} w/ step-VQA answer	52.55	37.29	67.81	64.38	48.85	79.90
4	syn_{at} w/ syn_{at} answer	53.75	37.92	69.58	65.10	48.85	81.35
5	syn_{at} w/ step-syn_{at} answer	55.21	40.21	70.21	66.41	51.35	81.46
6	syn_{at} w/ step-syn_{at} answer + tmpl.	57.34	42.60	72.08	67.86	53.12	82.60
7	syn_{at} w/ step-syn_{at} answer + tmpl. + addi.	58.65	45.62	71.67	69.84	56.56	83.12

Table 2. Main results on ChartQA val split. The data generator trained on **A**nnotated questions is denoted as syn_a; the data generator trained on both **A**nnotated questions and our **T**emplate-generated questions syn_{at} is denoted as syn_{at}. Step-by-step generation is better than straight-forward generation (row 1-5). Training with a combination of step-by-step generated QA, template-generated QA (+tmpl.), and additional images from chart captioning datasets (+addi.), leads to the best model. See Sec. 4.2 for more details.

For example, many questions in ChartQA ask about years (*e.g*. 1996, 2012), for which allowing 5% numerical errors (*i.e*. around 100 years) will result in incorrectly high results.

4.2. Main results

We study our data generation method with different settings on the ChartQA dataset, with MATCHA as our baseline. We first generate QA data using our method, and then use the generated data to train the MATCHA model. The strict accuracy and relaxed accuracy reported on the ChartQA validation split are shown in Tab. 2.

The baseline model MATCHA (row-0) achieves 47.76% accuracy and 58.54% relaxed accuracy. For this baseline, we finetune the pretrained MATCHA checkpoint for 10k iterations with the ChartQA training split (combing human and augmented questions), using 1024 image patches. This setting follows [35], except that a smaller number of image patches are used for faster experiments and less GPU memory requirement. We validate the correctness of our implementation by showing that the model reaches 63.7% relaxed accuracy using the original resolution, which matches the numbers reported in [35] (as shown in Tab. 3).

Row-1 shows the results for the straight-forward data generation, where we train the data generator with QAs from the ChartQA training split and use the trained generator to directly generate questions and answers. This straightforward data generation improves the accuracy by 1% (from 47.76% to 48.80%), suggesting that LLMs can be helpful in data augmentation for chart reasoning.

Row-2 to row-5 shows the advantage of synthesizing step-by-step over straight-forward generation. We train the data generator with our template-generated QA, but the answers are generated with different methods. In row-2, we feed in the generated complex question into an additional VQA model (*i.e*. baseline MATCHA) to get the answers; in row-3, we generate the step-by-step rationales, and the step-wise questions are fed into the VQA model to get step-wise answers, then the final answer is derived using the step-wise answers. As can be seen, the step-wise answer generation (52.55%) is significantly better than the direct answer generation (47.76%), which suggests the importance of decomposing the complex questions into atom sub-questions. Row-4 and row-5 follow a similar setting, except that we use the LLM data generator to generate the answers instead of the additional VQA model. Two findings can be drawn: (a) LLM-generated answers are better than VQA-predicted answers, possibly because the LLM data generator has access to the predicted data tables as inputs; (b) step-by-step synthesizing outperforms direct generation, for both VQA-predicted and LLM-generated answers.

In row-6, we combine the step-by-step generated data with the templated-generated QAs and show that training with both leads to further better performance (57.34%). In row-7, we generate QAs for the chart images from additional chart captioning datasets. The final model, trained with a combination of all the synthetic data, gets 58.65% accuracy. Compared with the baseline, training with the augmented data leads to a 10.89% accuracy improvement. Moreover, the improvement is more significant on the human-written questions (+15.41%), which requires complex reasoning, than the augmented extractive questions (+6.36%). The results suggest that our synthetic data improves both the extractive ability and the reasoning ability, with the latter being more significant. We show examples in the Appendix.

4.3. Comparison with SoTA

To compare our method with the best performing models, we apply our generated data onto two models, Pix2Struct [28] and MATCHA [35], and report the results on the ChartQA test split in Tab. 3. For faster training, we first train the model (for both Pix2struct and MATCHA) with a max of 1024 image patches using our best setting (*i.e*. row-7 in Tab. 2), then finetune with 4096 image patches to get the final result. We report both the 1024-resolution results and the 4096-resolution results.

	params	Accuracy avg	human	augment	Relaxed Accuracy avg	human	augment
VL-T5-OCR [40]	-	-	-	-	41.56	-	-
VisionTapas-OCR [40]	-	-	-	-	45.52	-	-
Pix2Struct$_{4096}$ [28]	282M	-	-	-	56.0	30.5	81.6
MatCha$_{4096}$ [35]	282M	-	-	-	64.2	38.2	90.2
DePlot+FlanPaLM+Codex [34]	540B	-	-	-	79.3	67.6	91.0
PaLI-X [7]	55B	-	-	-	72.3	-	-
Pix2Struct$_{4096}$ (reimpl.)	282M	50.44	24.64	**76.24**	58.24	32.00	**84.48**
Pix2Struct$_{1024}$ + LaMenda	282M	51.77	34.69	68.85	63.02	46.25	79.79
Pix2Struct$_{4096}$ + LaMenda	282M	**55.83**	**40.10**	71.56	**66.82**	**51.67**	81.98
MatCha$_{4096}$ (reimpl.)	282M	55.68	31.04	80.32	63.72	37.76	89.68
MatCha$_{1024}$ + LaMenda	282M	60.28	41.92	78.64	69.88	51.76	88.00
MatCha$_{4096}$ + LaMenda	282M	**63.36**	**44.88**	**81.84**	**72.64**	**53.92**	**91.36**

Table 3. Comparison with SoTAs on ChartQA test split. Subscripts ($_{1024}$, $_{4096}$) means image token numbers (resolution). With our generated data, both Pix2Struct and MatCha achieve significantly better performance, even comparable to much larger models.

As can be seen, our generated data improves the overall relaxed accuracy of Pix2Struct by 8.58% and MatCha by 8.92%. Moreover, the improvement is much more significant on human-written questions, *e.g.* the relaxed accuracy of MatCha on human-written questions improves from 38% to 54%. Interestingly, with our augmented data, the low-resolution (1024) models can already outperform their high-resolution (4096) counterparts without data augmentation. The significant performance gain indicates that learning with synthetic data is extremely helpful for better reasoning ability with human-written queries.

Our best performing MatCha achieves 72.6% relaxed accuracy, which beats all existing end-to-end chart reasoning models. Note that DePlot+FlanPaLM+Codex is a two-stage model that first extracts information from the charts, then prompts external LLMs like FlanPalm (540B) Codex (12B) for question answering. The method relies on much larger models of billions of parameters and thus is not comparable with our method, which is much smaller and not rely on external LLMs for inference.

4.4. Generalization

To show the generalization ability of the proposed data generation method, we run experiments on the PlotQA dataset. We directly use our data generator (trained on template questions on ChartQA) to generate QA for the PlotQA training images. The results are shown in Tab. 4. As can be seen, our method, when applied to MatCha, achieves state-of-the-art performance (92.89% relaxed accuracy) on PlotQA. Despite the synthetic nature of this dataset and the saturated model performance, our augmented data can still help on both low-resolution (+3.46%) and high-resolution (+0.92%) settings.

4.5. Ablations and analysis

Qualitative analysis and failure cases. Fig. 5 shows examples of our LLM-generated QA, including good and bad examples. Most generated questions are specific to the given image, asking about the categories plotted, such as "2019" or "Tablet 2018", indicating that the model is able to utilize the information of the input. Moreover, various questions are generated, including counting, colors, average, etc. In the failure example, the question requires comparing the "silver bar" with the "midnightblue" bar, with reasonable rationales. However, there are multiple bars in the specified colors in the image, resulting in ambiguous/invalid questions. When looking at more failure cases, we find that there are several typical failure modes. For example, questions are sometimes invalid/ambiguous (*e.g.* asking about the top/bottom bar for a vertical bar plot); questions can be hallucinated or irrelevant (*e.g.* asking about a category that is not shown in the figure); rationales and answers are not correct, etc., which opens up opportunities for future works.

Before and after LaMenda. To gain insights on how LaMenda helps model training, we conduct a comparison analysis of MatCha **"before" and "after"** training with our LLM-generated data, *i.e.* the baseline model versus our best model. According to the predictions of these two models, we divide the 960 human-written questions from ChartQA val split into 4 groups: both model answers correctly, both models answers wrongly, the before model answers correctly, but the after model answers wrongly, and

	Relaxed Accuracy avg	V1	V2
VisionTapas-OCR [40]	53.90	65.30	42.50
VL-T5-OCR [40]	65.96	75.90	56.02
DePlot+FlanPaLM+Codex [34]	66.6	62.2	71.0
Pix2Struct [28]	72.5	73.2	71.9
MatCha$_{4096}$ [35]	**91.5**	**92.3**	**90.7**
MatCha$_{1024}$ (reimpl.)	74.95	73.88	76.02
MatCha$_{1024}$ + LaMenda	78.41	78.44	78.38
MatCha$_{4096}$ (reimpl.)	91.97	92.64	91.30
MatCha$_{4096}$ + LaMenda	**92.89**	**93.94**	**91.84**

Table 4. Comparison with SoTAs on PlotQA test split. With our generated data, MatCha achieves the SoTA performance.

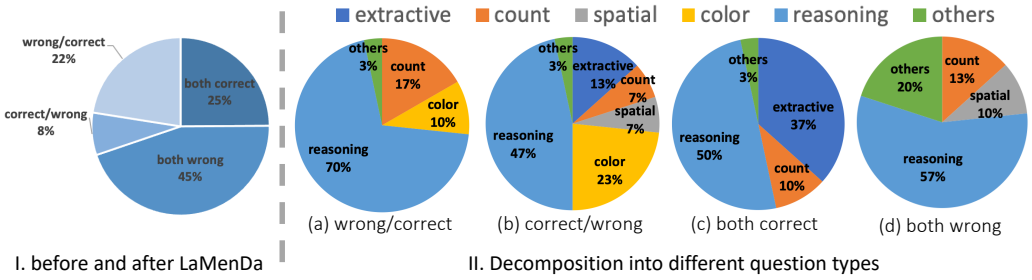


Figure 4. Analysis of model predictions **before** and **after** training with LaMenDa. In I (left), ChartQA val questions are divided into four categories according to the model prediction correctness (before/after). II (right) further decomposes each category into different question types. LaMenDa improves the model prediction, and the improvement is most significant on *reasoning* questions.

vice versa. The statistics of these four groups are shown in Fig. 4. From Fig. 4 I. (left side), we see that LaMenDa is helpful for 22% of the questions, while there are still 45% of the questions where both models cannot answer correctly. In Fig. 4 II. (right side), we randomly sample 30 questions from each of the four groups and manually classify them into six question types: extractive, count, spatial, color, reasoning, and others. As shown in (a), *i.e.* the questions where the "before" model answers wrongly and our model answers correctly, most (70%) of the questions belong to the reasoning category, indicating that our data helps most on the reasoning questions. Our data also helps with color questions (10%) and count questions (17%). However, our data does not help with extractive questions (0%). In addition, comparing the distribution of (c) both correct with (d) both wrong, we can see that extractive questions are easy for the models, while reasoning questions are very challenging.

In summary, Fig. 4 shows the effectiveness of our data for the **reasoning questions** over other question types. However, reasoning questions are still a big challenge for current models, which requires further exploration.

Different question types. In Tab. 5, we study the improvements brought by each type of the generated questions. The total 326k generated QAs are grouped into 7 different groups, including questions about color understanding, spatial reasoning, counting, minimum/maximum reasoning, average, comparison (*e.g.* larger than, smaller than), and other math calculations. This grouping is done by different prompting of the data generation LLM, as specified in the Appendix. The improvements from different question types are distinct and can be accumulated. As we see, each of the seven types improves the model performance by less than 3%, while combining all of them leads to 7.03% improvement. Moreover, among the seven types, the "math calculation" questions brings the most improvements (2.76%).

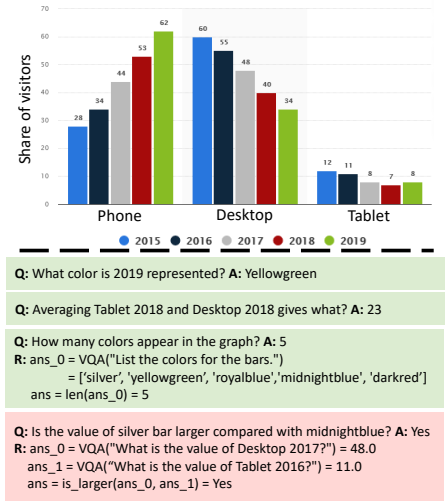


Figure 5. Examples of generated QA, 3 good examples and 1 failure case. Rationales are only shown for 2 examples for brevity.

	#QA	avg	human	aug
baseline	28,299	47.76	30.21	65.31
+colors	+46,512	47.81	29.90	65.73
+spatial	+49,387	49.27	30.63	67.92
+count	+36,705	47.40	29.17	65.63
+minmax	+71,788	49.58	31.87	67.29
+average	+50,717	48.96	31.35	66.56
+compare	+14,690	47.66	28.85	66.46
+calculation	+56,361	50.52	33.85	67.19
+all	326,160	54.79	39.27	70.31

Table 5. A detailed look at the effect for different question types. Strict accuracy on ChartQA val split is shown. Each type helps and combining all of them leads to a further performance gain.

5. Conclusion

We propose Synthesize Step-by-Step, which utilizes LLMs to augment the training data for the chart reasoning task. We show that the step-by-step generation, which decomposes the complex reasoning questions into easier sub-questions, is critical for synthesizing good-quality data. We run experiments on two standard benchmarks and show significant improvements by training with the augmented data. Specifically, the relaxed accuracy on the human-written questions is improved from 37.8% to 53.9% on ChartQA. The large improvement indicates the promise of boosting the model's reasoning ability by data generation using LLMs, from which the strong step-by-step reasoning ability can be distilled into downstream models.

References

[1] Mubashara Akhtar, Oana Cocarascu, and Elena Simperl. Reading and reasoning over chart images for evidence-based automated fact-checking. *arXiv preprint arXiv:2301.11843*, 2023. 2

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022. 2

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016. 2

[4] Srikar Appalaraju, Peng Tang, Qi Dong, Nishant Sankaran, Yichu Zhou, and R Manmatha. Docformerv2: Local features for document understanding. *arXiv preprint arXiv:2306.01733*, 2023. 2

[5] Ritwick Chaudhry, Sumit Shekhar, Utkarsh Gupta, Pranav Maneriker, Prann Bansal, and Ajay Joshi. Leaf-qa: Locate, encode & attend for figure question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3512–3521, 2020. 2

[6] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020. 2

[7] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023. 2, 7

[8] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022. 2

[9] John Joon Young Chung, Ece Kamar, and Saleema Amershi. Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions. *arXiv preprint arXiv:2306.04140*, 2023. 2

[10] Brian Davis, Bryan Morse, Brian Price, Chris Tensmeyer, Curtis Wigington, and Vlad Morariu. End-to-end document recognition and understanding with dessurt. In *European Conference on Computer Vision*, pages 280–296. Springer, 2022. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 2

[13] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023. 2

[14] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023. 2

[15] Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. *arXiv preprint arXiv:1912.04971*, 2019. 2

[16] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023. 2, 12

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[18] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023. 2

[19] Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018. 2

[20] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019. 4

[21] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. 4, 12

[22] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. Dvqa: Understanding data visualizations via question answering, 2018. 2

[23] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Akos Kadar, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning, 2018. 2

[24] Shankar Kantharaj, Rixie Tiffany Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. Chart-to-text: A large-scale benchmark for chart summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4005–4023, 2022. 5, 14

[25] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022. 2

[26] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon

Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *European Conference on Computer Vision*, pages 498–517. Springer, 2022. 2

[27] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022. 2

[28] Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR, 2023. 2, 5, 6, 7, 15

[29] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023. 2

[30] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 2

[31] Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. Symbolic chain-of-thought distillation: Small models can also" think" step-by-step. *arXiv preprint arXiv:2306.14050*, 2023. 2

[32] Zhuowan Li, Elias Stengel-Eskin, Yixiao Zhang, Cihang Xie, Quan Hung Tran, Benjamin Van Durme, and Alan Yuille. Calibrating concepts and operations: Towards symbolic reasoning on real images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14910–14919, 2021. 2

[33] Zhuowan Li, Xingrui Wang, Elias Stengel-Eskin, Adam Kortylewski, Wufei Ma, Benjamin Van Durme, and Alan L Yuille. Super-clevr: A virtual benchmark to diagnose domain robustness in visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14963–14973, 2023. 2, 4

[34] Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. Deplot: One-shot visual language reasoning by plot-to-table translation. *arXiv preprint arXiv:2212.10505*, 2022. 2, 3, 5, 7, 15

[35] Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *arXiv preprint arXiv:2212.09662*, 2022. 1, 2, 5, 6, 7, 15

[36] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 2, 3, 5

[37] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*, 2023. 2

[38] Junyu Luo, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. Chartocr: Data extraction from charts images via a deep hybrid framework. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1917–1925, 2021. 2

[39] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019. 2

[40] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022. 1, 2, 5, 7, 15

[41] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536, 2020. 2, 5

[42] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022. 2

[43] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. 2

[44] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. 2

[45] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023. 2

[46] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 3

[47] Raian Rahman, Rizvi Hasan, Abdullah Al Farhad, Md Tahmid Rahman Laskar, Md Hamjajul Ashmafee, and Abu Raihan Mostofa Kamal. Chartsumm: A comprehensive benchmark for automatic chart summarization of long and short summaries. *arXiv preprint arXiv:2304.13620*, 2023. 5, 14

[48] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. 2

[49] Timo Schick and Hinrich Schütze. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, 2021. 2

[50] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv*

preprint arXiv:2303.17580, 2023. 2

[51] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023. 2

[52] Benny J Tang, Angie Boggust, and Arvind Satyanarayan. Vistext: A benchmark for semantically rich chart captioning. *arXiv preprint arXiv:2307.05356*, 2023. 5, 14

[53] Peng Tang, Srikar Appalaraju, R Manmatha, Yusheng Xie, and Vijay Mahadevan. Multiple-question multiple-answer text-vqa. *arXiv preprint arXiv:2311.08622*, 2023. 2

[54] Peng Tang, Pengkai Zhu, Tian Li, Srikar Appalaraju, Vijay Mahadevan, and R Manmatha. Deed: Dynamic early exit on decoder for accelerating encoder-decoder transformer models. *arXiv preprint arXiv:2311.08623*, 2023. 2

[55] MN Team et al. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. 3

[56] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021. 2

[57] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help. *arXiv preprint arXiv:2108.13487*, 2021. 2

[58] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. 2

[59] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022. 2

[60] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. *arXiv preprint arXiv:2110.07178*, 2021. 2

[61] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200, 2020. 2

[62] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023. 2

[63] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023. 2

[64] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018. 2

[65] Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. Large language model as attributed training data generator: A tale of diversity and bias. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. 2

[66] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022. 2

[67] Yanzhe Zhang, Ruiyi Zhang, Jiuxiang Gu, Yufan Zhou, Nedim Lipka, Diyi Yang, and Tong Sun. Llavar: Enhanced visual instruction tuning for text-rich image understanding. *arXiv preprint arXiv:2306.17107*, 2023. 2

[68] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022. 2

[69] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023. 2

[70] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 2

A. Data scaling analysis

To analyze the influence of data amount, we train MATCHA using different amount (25%, 50%, 75%, 100%) of our LAMENDA data, and report results in the Tab. 6. We use the same setting and data as row-5 in Tab-2, except that the training schedule is shorter (5k iterations) due to time limitations. As shown, improvements of our generated data follow a typical data scaling law: improvements are large initially (at 25%) then continues with smaller margins.

data amount	0%	25%	50%	75%	100%
human	30.21	34.38	35.73	36.25	37.19
augment	65.31	66.87	68.85	69.69	68.96
avg.	47.76	50.63	52.29	52.97	53.07

Table 6. Results with different amount of generated data.

B. Template-based QA generation

Results using template-generated data only. We report the result using template-generated QA data only, under the same setting as Tab-2 (1024 patches, on ChartQA): training with only template QAs leads to 54.32% strict accuracy (38.23% on human, 70.73% on augmented). This result is better than baseline (47.76%), but lower than LLM generator which gets 58.65%. Additionally, although template QAs are very clean, they (a) are not free-form and (b) rely on groundtruth SVG metadata, thus cannot generalize to images without SVG, which highlights the advantages of LLM generator.

Template list. Empirically, we find that template diversity is crucial for the model performance, otherwise the LLM generator may overfit to rigid templates. For example, in the early stage of the project, our LLM generator gets only 49.8% accuracy trained with 16 templates without rephrasing, which is much lower than the current 55.2% (row 5 Tab-2). Tab. 7 shows the final list of templates that are used for the templated-based question-answer generation pipeline. For each template, we manually define a functional program as inspired by the CLEVR dataset [21]. The functions include 7 basic operations like `SUM`, `COUNT`, `COMPARE`, and a `VQA` operation. The execution is motivated by VisProg [16], where each operation is executed by a predefined Python function.

C. Prompts

Tab. 8 shows the prompts to prompt the LLM-based data generator, for controllably generating questions and answers.

D. Dataset statistics

Tab. 9 shows detailed statistics for the ChartQA and the PlotQA datasets. Tab. 10 shows the detailed statistics of the chart captioning datasets. We generated questions and answers using our LLM-based data generator for the chart images in these chart captioning datasets.

E. Additional results

Tab. 11 shows the full results including relaxed accuracy and strict accuracy for Tab. 4 in the main paper. Tab. 12 shows the full results including relaxed accuracy and strict accuracy for Tab. 5 in the main paper.

F. Examples comparing MATCHA with and without LAMENDA

Color
1. What color is $\langle N \rangle$ represented?
2. What is the value of the $\langle C \rangle \langle F \rangle$?
3. Which category is represented by $\langle C \rangle$?
Spatial
4. What is the value of the $\langle S \rangle$ bar?
5. What does the $\langle S \rangle$ bar represent?
6. What is the value of the second bar from the $\langle S \rangle$?
7. What is [represented by] the second bar from the $\langle S \rangle$?
8. What is the value of the third bar from the $\langle S \rangle$?
9. What is represented by the third bar from the $\langle S \rangle$?
Count
10. How many $\langle F \rangle$s are shown in the plot?
11. How many $\langle C \rangle$ bars are shown in the plot?
12. How many colors are used to represent the $\langle F \rangle$s in the plot?
Math
13. What is the average of $\langle N \rangle$?
14. What is the max [value] of $\langle N \rangle$?
15. What is the min value of the $\langle N \rangle$?
16. What is the [total] sum [value] of $\langle L \rangle$ and $\langle L2 \rangle$?
17. What is the difference between [values of] $\langle L \rangle$ and $\langle L2 \rangle$?
18. What is the value of the smallest category[in the chart]?
19. What is the value of the largest category[in the chart]?
20. What is the smallest category?
21. What is the largest category?
22. What is the average [value] of the two smallest categories[in the chart]?
23. What is the average [value] of the two largest categories[in the chart]?
24. What is the difference between the largest [category] and the smallest category?
25. What is the ratio [value] of $\langle L \rangle$ and $\langle L2 \rangle$?
26. How many times [is] $\langle L \rangle$ bigger than $\langle L2 \rangle$?
27. What is the average of $\langle L \rangle$ and $\langle L2 \rangle$?
28. Is [the value of] $\langle L \rangle$ more than $\langle L2 \rangle$?

Table 7. List of templates.

1. The question should be similar to this: ...
2. The question should be free form.
3. The question should require color understanding of the image.
4. The question should require counting.
5. The question should require counting of colors.
6. The question should require counting and color understanding.
7. The question should require spatial understanding of the image.
8. The question should require math reasoning about min.
9. The question should require math reasoning to compute min.
10. The question should require math reasoning to compute average of two categories.
11. The question should require math reasoning to compute average.
12. The question should require math reasoning to compute max.
13. The question should require math reasoning about the difference between max and min.
14. The question should require math reasoning to compute difference.
15. The question should require math reasoning about comparison.
16. The question should require math reasoning about average and max.
17. The question should require math reasoning to compute sum.
18. The question should require math reasoning about max.
19. The question should require math reasoning about average and min.
20. The question should require math reasoning to compute ratio.
21. The question should require color understanding and math reasoning to compute difference.
22. The question should require color understanding and math reasoning about comparison.
23. The question should require spatial understanding and math reasoning to compute difference.
24. The question should require spatial understanding and math reasoning about average.

Table 8. List of prompts for prompting the LLM-based data generator.

	ChartQA			**PlotQA**		
	Images	**HumanQA**	**AugmentedQA**	**Images**	**V1 QA**	**V2 QA**
train	18,317	7,398	20,901	157,070	5,733,893	20,249,479
val	1,056	960	960	33,653	1,228,468	4,360,648
test	1,509	1,250	1,250	33,660	1,228,313	4,342,514

Table 9. Statistics for ChartQA and PlotQA.

		#images	#captions	#llava_pred	#filter-10
Chart-to-text [24]	statista_two_col	27868	27868	603283	613096
	statista_multi_col	6943	6943	152746	107752
	pew_two_col	1486	1486	31806	23521
	pew_multi_col	7799	7799	171578	113967
VisText [52]	vistext	8822	9969	172319	76788
ChartSumm [47]	img_list_s	40985	32786	901670	364218
	img_list_k	43378	34702	954316	336022
All	-	137281	121553	2987718	1635364

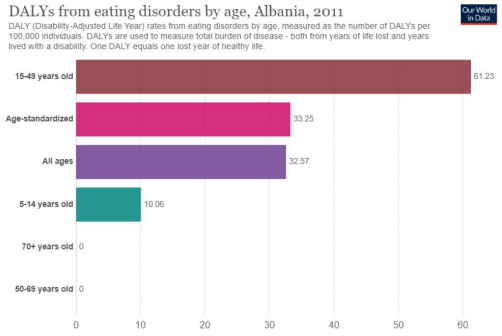
Table 10. Statistics for the chart captioning datasets.

	Accuracy			Relaxed Accuracy		
	avg	**V1**	**V2**	**avg**	**V1**	**V2**
VisionTapas-OCR [40]	-	-	-	53.90	65.30	42.50
VL-T5-OCR [40]	-	-	-	65.96	75.90	56.02
DEPLOT+FlanPaLM(540B)+Codex [34]	-	-	-	66.6	62.2	71.0
Pix2Struct [28]	-	-	-	72.5	73.2	71.9
MATCHA [35]	-	-	-	**91.5**	**92.3**	**90.7**
MATCHA_{1024} (reimpl.)	41.58	66.66	16.50	74.95	73.88	76.02
MATCHA_{1024} + LAMENDA	43.04	68.48	17.60	78.41	78.44	78.38
MATCHA_{4096} (reimpl.)	50.89	76.14	25.64	91.97	92.64	91.30
MATCHA_{4096} + LAMENDA	**51.40**	**76.42**	**26.38**	**92.89**	**93.94**	**91.84**

Table 11. Full results for Tab. 4: comparison with SoTAs on PlotQA test split. With our generated data, MATCHA achieves the SoTA performance.

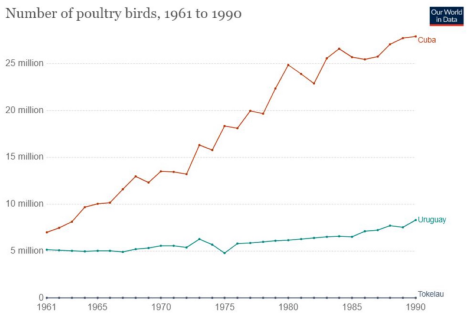
	# Questions	Accuracy			Relaxed Accuracy		
		avg	**human**	**augment**	**avg**	**human**	**augment**
baseline	28,299	47.76	30.21	65.31	58.54	39.58	77.50
+colors	+46,512	47.81	29.90	65.73	59.32	40.21	78.44
+spatial	+49,387	49.27	30.63	67.92	61.09	43.33	78.85
+count	+36,705	47.40	29.17	65.63	58.28	38.54	78.02
+minmax	+71,788	49.58	31.87	67.29	59.38	40.63	78.13
+average	+50,717	48.96	31.35	66.56	60.00	40.52	79.48
+compare	+14,690	47.66	28.85	66.46	58.59	38.85	78.33
+calculation	+56,361	50.52	33.85	67.19	62.40	45.21	79.58
+all	+326,160	54.79	39.27	70.31	66.15	50.42	81.88

Table 12. Full results for Tab. 5: a detailed look at the effect for different question types. Strict accuracy on ChartQA val split is shown. Each type helps and combining all of them leads to a further performance gain.



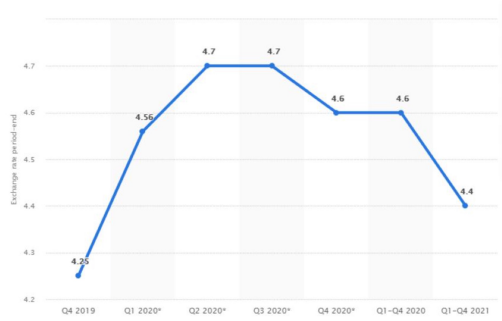
Question : How many age groups are shown in the graph?
GT Answer : 6
MATCHA$_{baseline.}$: 7
MATCHA$_{LAMENDA}$: 6

(a)



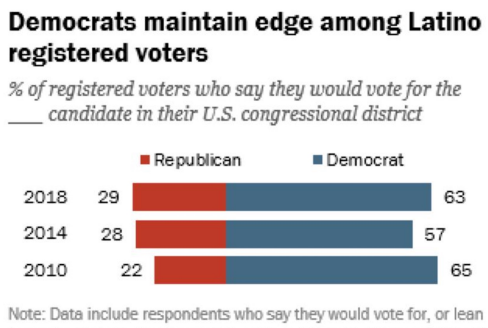
Question : In which year, the middle line (Uruguay) is lowest?
GT Answer : 1975
MATCHA$_{baseline.}$: 1961
MATCHA$_{LAMENDA}$: 1975

(b)



Question: What is the third data value in the blue bar? left to right
GT Answer : 4.7
MATCHA$_{baseline.}$: 4.6
MATCHA$_{LAMENDA}$: 4.7

(c)



Question: What is the total of Republicans and Democrats in 2010?
GT Answer : 87
MATCHA$_{baseline.}$: 113
MATCHA$_{LAMENDA}$: 87

(d)

Figure 6. The baseline MatCha fails to correctly answer the question which needs visual understanding and then doing arithmetic operation, whereas MatCha fine-tuned with our data is able to correctly answer it.