UCLA
UCLA Electronic Theses and Dissertations

Title

Advancing Mathematical Reasoning with Language Models: A Multimodal and Knowledge-Intensive Perspective

Permalink

https://escholarship.org/uc/item/678864d8

Author

Lu, Pan

Publication Date

2024

Peer reviewed|Thesis/dissertation

eScholarship.org
Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
Los Angeles

Advancing Mathematical Reasoning with Language Models: A Multimodal and Knowledge-Intensive Perspective

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Pan Lu

2024

© Copyright by

Pan Lu

2024

ABSTRACT OF THE DISSERTATION

Advancing Mathematical Reasoning with Language Models: A Multimodal and Knowledge-Intensive Perspective

by

Pan Lu
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Kai-Wei Chang, Co-Chair
Professor Song-Chun Zhu, Co-Chair

Mathematical reasoning is a pivotal component of human intelligence, crucial for advancing education and science. This dissertation delves into the development of language model systems capable of robust mathematical reasoning, marking a significant step toward realizing general artificial intelligence. We introduce multi-modal and knowledge-intensive benchmarks to assess the reasoning capabilities of large language models (LLMs) and vision-language models (VLMs) across real-world contexts, including visual information, tabular data, and scientific domains.

This dissertation advances the field by proposing new pre-trained VLMs. For instance, PatchTrm introduces a patch-based cross-modal Transformer model for abstract diagram reasoning. We also present innovative retrieval and tool-augmented algorithms that enhance LLM capabilities. Notably, Inter-GPS is a neuro-symbolic solver for geometry that demonstrates human-level performance, marking a first in the domain. Additionally, PromptPG pioneers the use of reinforcement learning for dynamic in-context example selection, significantly improving the stability and accuracy of LLMs. Another groundbreaking contribution is Chameleon, a model that integrates LLMs with external tools, vastly increasing their flexibility and effectiveness in real-world applications. The dissertation concludes by analyzing the latest advances in mathematical reasoning within visual contexts, and highlighting the current challenges and future prospects.

The dissertation of Pan Lu is approved.

Ying Nian Wu

Nanyun Peng

Guy Van den Broeck

Song-Chun Zhu, Committee Co-Chair

Kai-Wei Chang, Committee Co-Chair

University of California, Los Angeles

2024

To my family and my love

who taught me to

dream,

persevere, and

be grateful.

TABLE OF CONTENTS

1 Introduction **1**

1.1 Motivation 1

1.2 Dissertation Outline 4

I Multimodal and Knowledge-Intensive Benchmarks **7**

2 Multimodal Mathematical Reasoning **8**

2.1 Geometry Problem Solving 8

2.1.1 Introduction 8

2.1.2 Dataset Collection 8

2.1.3 Dataset Statistics 10

2.1.4 Comparisons with Existing Datasets 11

2.1.5 Human Performance 12

2.2 Math Word Problems in Abstract Scenes 12

2.2.1 Introduction 12

2.2.2 Dataset Collection 14

2.2.3 Data Analysis 15

2.3 Math Word Problems in Tabular Contexts 19

2.3.1 Introduction 19

2.3.2 Related Datasets 21

2.3.3 Dataset Collection 21

2.3.4 Dataset Statistics 23

3 Knowledge-Intensive Scientific Reasoning **25**

3.1 Introduction 25
3.2 Related Work 27
3.3 Dataset 29
3.3.1 Data Collection 29
3.3.2 Data Analysis 30
3.3.3 Choice Statistics 34
3.3.4 Grade Statistics 34
3.3.5 Comparisons with Existing Datasets 35
3.4 Baselines and Chain-of-Thought Models 36
3.4.1 Heuristic Baselines 36
3.4.2 Language Model Baselines 37
3.4.3 Language Models with the Chain of Thought 38
3.4.4 Experimental Setup 39
3.4.5 Results for Question Answering 40
3.4.6 Human Evaluation of Generated Explanations 43
3.4.7 Analysis 44
3.4.8 Case Study 47
3.5 Conclusion 49

II Pre-trained Vision-Language Models 51

4 Patch Cross-modal Transformer Model 52
4.1 Introduction 52
4.2 Related Work 54
4.3 Patch Cross-modal Transformers 55

4.3.1 Diagram Encoder 56

4.3.2 Language Encoder 57

4.3.3 Answer Reasoning 57

4.4 Image Encoder Pre-training 58

4.4.1 Pre-training Data: Icon645 59

4.4.2 Icon Classification for Pre-training 60

4.5 Experiments 60

4.5.1 Baselines 61

4.5.2 Experimental Details 62

4.5.3 Experimental Results 63

4.5.4 Ablation Study 65

4.5.5 Quantitative Analysis 66

4.6 Conclusion 66

4.7 Appendix: User Study 67

4.7.1 Crowd Sourcing Method 67

4.7.2 Quality Assurance 68

III Retrieval and Tool-Augmented Algorithms 70

5 Neuro-Symbolic Geometric Solver **71**

5.1 Introduction 71

5.2 Related Work 73

5.3 Geometry Formal Language 74

5.4 Geometry Problem Parser 74

5.4.1 Text Parser 75

5.4.2 Diagram Parser 75
5.5 Geometry Problem Solver 76
5.5.1 Symbolic Geometry Solver 77
5.5.2 Theorem Predictor (TP) 77
5.5.3 Low-first Search Strategy 78
5.6 Experiments 80
5.6.1 Experimental Settings 80
5.6.2 Comparisons with Baselines 81
5.6.3 Ablation Study and Discussion 82
5.7 Conclusion 84
5.8 Appendix: Gemetric Formal Language 85

6 Automatic Demonstration Learning 88
6.1 Introduction 88
6.2 Related Work 90
6.2.1 Prompt Learning for Language Models 90
6.2.2 Policy Gradient 90
6.3 Dynamic Prompting via Policy Gradient 91
6.4 Experiments 93
6.4.1 Evaluation Task 93
6.4.2 Evaluation Baselines 94
6.4.3 Evaluation metric 96
6.4.4 Implementation details. 96
6.4.5 Experimental Results 97
6.4.6 Ablation Study 99

6.4.7 Case Study . . . 101

7 Tool-Augmented Compositional Reasoning . . . 106

7.1 Introduction . . . 106

7.2 Related Work . . . 109

7.3 General Framework: Chameleon . . . 111

7.4 Applications of Chameleon . . . 113

7.4.1 Module Inventory . . . 113

7.4.2 Science Question Answering . . . 115

7.4.3 Tabular Mathematical Reasoning . . . 116

7.5 Experiments . . . 116

7.5.1 Experimental Details . . . 116

7.5.2 Experimental Results . . . 118

7.5.3 Qualitative Analysis . . . 121

7.5.4 Case Study . . . 123

7.5.5 Error Analysis . . . 127

7.6 Conclusion . . . 128

7.7 Appendix . . . 128

IV Summary and Future Directions 139

8 Latest Advances in Visual Mathematical Reasoning . . . 140

8.1 Understanding Mathematical Reasoning in Visual Contexts . . . 140

8.2 The MathVista Benchmark . . . 142

8.2.1 Collection Guidelines . . . 142

8.2.2 Benchmark Statistics . . . 146

8.3 Vision-Language Models in MathVista 146
8.3.1 Evaluation Methodologies 146
8.3.2 Experimental Setup 147
8.4 Tool-Augmented LLMs in MathVista 151
8.5 Exploring Human-AI Collaborations 153

9 Future Work and Conclusion 156
9.1 Current Challenges and Future Prospects 156
9.1.1 Analyzing Performance Saturation and Bridging Performance Gaps 156
9.1.2 Strategies for Self-Improvement 157
9.1.3 Retrieval and Tool-augmented Algorithms 158
9.1.4 Scientific Discovery and Autonomous Scientists 159
9.2 Conclusion 159

References 161

LIST OF FIGURES

1.1 A geometry problem requiring diagram understanding, theorem selection, and step-by-step calculation. 2

1.2 A science question requiring multimodal understanding, knowledge retrieval, and multi-hop reasoning. 2

1.3 Math word problems in tabular context requiring knowledge retrieval and tool utilization. 3

2.1 More data examples in the Geometry3K dataset. 9

2.2 Question length distribution of Geometry3K. 10

2.3 Examples from the IconQA dataset. Top: *multi-image-choice* sub-task. Middle: *multi-text-choice* sub-task. Bottom: *filling-in-the-blank* sub-task. 14

2.4 Question statistics based on number of words in IconQA. 16

2.5 Top 40 icons mentioned in the IconQA question texts and their appearance percentage. These icons cover various types of real-world objects. 16

2.6 Question types in IconQA. 17

2.7 Word cloud of the question text in IconQA. 17

2.8 Skill distribution in IconQA questions. 18

2.9 Two examples from the TabMWP dataset. The top is a *free-text* problem with a numerical answer; the bottom is a *multi-choice* problem with a textual answer. . . . 20

3.1 We construct the ScienceQA dataset where a data example consists of multimodal question answering information and the grounded lecture and explanation. 26

3.2 Question distribution of the ScienceQA dataset. 31

3.3 Question length distribution of related datasets in ScienceQA. 32

3.4 Word cloud distributions of question texts in different subjects in ScienceQA. . . . 33

3.5 Question distribution with different context formats in ScienceQA. 33

3.6 Domain diversity in ScienceQA. Each color corresponds to one subject: natural science, social science, and language science. 34

3.7 Choice length distribution in ScienceQA. 35

3.8 Grade distribution statistics in ScienceQA. 35

3.9 Instructions for AMT workers to answer the ScienceQA questions. 37

3.10 A test question example from ScienceQA presented to AMT workers for evaluation. 37

3.11 Prompt instruction encoding for the test example t in GPT-3 (CoT). 39

3.12 Interface of instructions for AMT workers to evaluate the explanations generated from UnifiedQA (CoT) and GPT-3 (CoT). 42

3.13 One example of the predicted answer along with the chain of thought from GPT-3 (CoT). 43

3.14 Accuracy of GPT-3 (CoT) cross different prompt types with 4-shot examples. . . . 44

3.15 Accuracy of GPT-3 (CoT) cross different numbers of training examples (b). 45

3.16 UnifiedQA (CoT) learns efficiently with fewer training examples. 46

3.17 A natural science example with the correct answer and a gold explanation. 47

3.18 A social science example with a correct answer and a gold explanation. 47

3.19 A language science example with a correct answer and a gold explanation. 48

3.20 An example with a correct answer but an irrelevant explanation. 48

3.21 An example with a correct answer but an incorrect explanation. 49

3.22 An example with a correct answer but an incomplete explanation. 49

3.23 An example with a wrong answer and a wrong explanation. 49

3.24 An example with a wrong answer and a wrong explanation. 50

3.25 An example with a wrong answer and a wrong explanation. 50

4.1 Top: Examples in three popular VQA datasets: VQA [GKS17], VQA 2.0 [GKS17], and CLEVR [JHM17]. Bottom: Examples of three sub-tasks in our collected IconQA dataset. 53

4.2 Our proposed model Patch-TRM for the IconQA task. 56

4.3 An overview of benchmark baselines on the IconQA task. 60

4.4 Performance of humans in different age groups for the IconQA task. Left: Accuracy over three sub-tasks; Right: Accuracy over thirteen reasoning skills. 64

4.5 An example of text-to-image attention visualization by Patch-Trm. 66

4.6 Result examples predicted by Patch-TRM in the IconQA test set. Top: *Multi-image-choice* sub-task. Middle: *Multi-text-choice* sub-task. Bottom: *Filling-in-the-blank* sub-task. Correctly predicted answers are highlighted in green, while wrong ones are highlighted in red. 67

4.7 Instructions provided to AMT workers for the user study of the IconQA dataset. . . 68

4.8 Attention check questions presented to AMT workers during the IconQA user study. 69

4.9 An AMT example question from the *multi-image-choice* sub-task of IconQA. . . . 69

5.1 A data example from the Geometry3K dataset, annotated with formal language descriptions for both the problem text and diagram. 72

5.2 Overview of the Inter-GPS architecture. Given the problem diagram and text, Inter-GPS first parses the inputs into a relation set defined in formal language. Then it applies the theorem sequence predicted by the theorem predictor to perform symbolic reasoning over the relation set to infer the answer. ⑨ and ⑯ denote the orders of the applied theorems. 76

5.3 Distribution of correctly solved problems by the number of search steps required by Inter-GPS. 83

5.4 Examples of failure cases encountered by Inter-GPS on the Geometry3K dataset. . 84

6.1 Overview of our proposed PromptPG approach, which learns to select high-performing in-context examples via policy gradient by interacting with the GPT-3 API, without relying on any manually designed heuristics. 89

6.2 User interfaces for the human study on the TabMWP dataset, showcasing the design for both *free-text* and *multi-choice* question types. 96

6.3 Accuracy of PromptPG with respect to different numbers of training and candidate examples. Experiments are conducted on 1,000 development instances from TabMWP, with each setting repeated using four random seeds to ensure robustness. 100

6.4 Two in-context examples selected by PromptPG, the prompt, and the correct prediction. The selected examples require similar abilities of mathematical reasoning to the test example. 103

6.5 In-context examples selected by *nearest neighbor search*, the prompt, and the incorrect prediction. The selected examples are semantically similar to the test example. 104

6.6 *Eandomly* selected in-context examples, the prompt, and the incorrect prediction. The selected examples have limited relevance to the test example,. 105

7.1 Examples of our Chameleon approach with GPT-4 on ScienceQA, a multi-modal question answering benchmark in scientific domains. Chameleon adapts to different queries by synthesizing programs that compose various tools and execute them sequentially to generate final answers. 107

7.2 Results of main baselines and Chameleon. Dashed lines represent human performance. 121

7.3 Distribution of tools called in the programs generated by Chameleon on ScienceQA. 122

7.4 Distribution of tools called in the programs generated by Chameleon on TabMWP. 122

7.5 Transition probabilities between modules in programs generated by Chameleon (GPT-4) on ScienceQA. START is the start symbol, END is a terminal symbol, and the others are non-terminal symbols. 123

7.6 Transition probabilities between modules in programs generated by Chameleon (GPT-4) on TabMWP. START is the start symbol, END is a terminal symbol, and the others are non-terminal symbols. 124

7.7 One more example from our Chameleon (GPT-4) approach on ScienceQA. 125

7.8 Three examples of our Chameleon approach with GPT-4 on TabMWP, a mathematical reasoning benchmark with tabular contexts. 126

7.9 Distribution of mistake examples across different categories on ScienceQA. Image: image captioning, Knowledge: knowledge understanding, Solution: solution generation. 127

8.1 Examples of seven mathematical reasoning categories in MathVista. 144

8.2 Examples of different visual contexts in MathVista. 145

8.3 Examples of our newly annotated datasets: IQTest, FunctionQA, and PaperQA. . . 145

8.4 Source dataset distribution of MathVista. 148

8.5 Accuracies of one leading LLM (*i.e.*, PoT GPT-4), three prominent VLMs, random chance, and human performance on our proposed MathVista across mathematical reasoning and visual context types. 150

8.6 Most AI models without tool augmentation struggle to solve this complex mathematical reasoning example. 151

8.7 Two examples showcasing GPT-4's performance on MathVista, which depends on the quality of generated captions and detected OCR texts. 152

8.8 GPT-4V initially produces an incorrect solution due to visual perception errors but recalibrates after receiving user feedback. 154

8.9 GPT-4V correctly identifies the number sequence but struggles with the underlying pattern until receiving user feedback. 155

LIST OF TABLES

2.1 Basic statistics of our Geometry3K dataset. 10

2.2 Most and least frequent predicates in the formal descriptions of the Geometry3K dataset (with a frequency greater than 5). 11

2.3 Comparison of our Geometry3K dataset with existing datasets. 11

2.4 Trigger words in metadata used to categorize skills in the IconQA dataset. 15

2.5 Statistics for the IconQA dataset. 16

2.6 Definition of reasoning skill types the in IconQA dataset. 17

2.7 Skill numbers for questions in the IconQA dataset. 18

2.8 Statistics for the IconQA dataset and comparisons with existing datasets. 19

2.9 Three different formats for the tables in the TabMWP dataset. 22

2.10 Format diversity of questions and answers in the TabMWP dataset. 23

2.11 Key statistics for the TabMWP dataset. 23

2.12 A comparison of MWP and Table QA datasets that require numerical reasoning. *text**: each table in TabMWP is accompanied by an image format. 24

3.1 Main statistics of the ScienceQA dataset. 30

3.2 Choice number distribution in ScienceQA. 34

3.3 Statistics for the ScienceQA dataset and comparisons with existing datasets. 36

3.4 Evaluation of baselines over different classes in accuracy (%). Model names: Q = question, M = multiple options, C = context, C_T = text context, C_I = image context, CoT = chain of thought. Format names: A = answer, AE = answer with explanation, ALE = answer with lecture and explanation. Question classes: NAT = natural science, SOC = social science, LAN = language science, TXT = text context, IMG = image context, G1-6 = grades 1-6, G7-12 = grades 7-12. 41

3.5 Automatic metrics (BLEU-1/4, ROUGE-L, Similarity) and human evaluation of generated explanations. A gold explanation refers to one that is relevant, correct, and complete. 44
3.6 Dynamic sampling for GPT-3 (CoT). 45
3.7 Upper bound of GPT-3 (CoT). 46
3.8 Different positions of L/E for GPT-3 (CoT). 46

4.1 Statistics for the Icon645 dataset. 59
4.2 Collected icon examples in the Icon645 dataset. 59
4.3 Results for icon classification. 60
4.4 Results on the IconQA dataset. 64
4.5 Human performance in the IconQA task. 64
4.6 Ablation study in the IconQA dataset. 66

5.1 Examples of terms used in the formal language for geometry problem solving. . . . 74
5.2 Evaluation results of our proposed method and comparison with baselines on the Geometry3K dataset. 81
5.3 Evaluation results on the GEOS dataset. 82
5.4 Performance of Inter-GPS with different search strategies. 82
5.5 Performance of Inter-GPS with predicted and ground truth (GT) literals. 83
5.6 Neural solver performance with different representations of the problem text and diagrams. 84
5.7 20 predicates and corresponding literal templates for geometric shapes. 85
5.8 9 predicates and corresponding literal templates for unary geometric attributes. . . 86
5.9 17 predicates and corresponding literal templates for general geometric attributes . . 86
5.10 12 predicates and corresponding literal templates for binary geometric relations. . . 86

5.11 15 predicates and corresponding literal templates for A-IsXOf-B-type geometric relations. 87

5.12 18 predicates and corresponding literal templates for numerical attributes and relations. 87

6.1 Evaluation results of various baselines and our method on TabMWP. 98

6.2 Blind studies on TabMWP. T: tabular context; Q: question; C: choice options; A: answer. Q(C) means choice options follow the question in the input, while Q refers to the question only. 99

6.3 Evaluation results of different selection strategies with three trials. 101

6.4 Results of different numbers of few-shot examples on 1,000 development examples. 102

7.1 A comparison of work that augments large language models with tool usage. . . . 110

7.2 Different tools in our module inventory. 113

7.3 Tools used on ScienceQA and TabMWP. Reusable tools are marked in green. . . . 116

7.4 QA accuracy (%) on the test set of ScienceQA. 119

7.5 QA accuracy (%) on the test set of TabMWP. 120

7.6 Score drop with disabled modules on ScienceQA and TabMWP. 122

7.7 The statistics of the number of different generated programs and the average length of generated programs by Chameleon, respectively. 125

7.8 The prompt constructed for the planner model on the ScienceQA task. The prompt consists of the instruction that describes the role of the planner model, the in-context examples that map the problem to the module sequence, and the test example. 129

7.9 The prompt constructed for the planner model on the TabMWP task. Similarly, the prompt consists of the instruction, the in-context examples, and the test example. . 130

7.10 The prompt constructed for the "Knowledge Retrieval" module on the ScienceQA task. 131

7.11 The prompt constructed for the "Query Generator" module on the ScienceQA task. 131

7.12 The prompt constructed for the "Solution Generator" module on the ScienceQA task. 132

7.13 The prompt constructed for the "Knowledge Retrieval" module on the TabMWP task. 133

7.14 The prompt constructed for the "Row Lookup" module on the TabMWP task. . . . 133

7.15 The prompt constructed for the "Column Lookup" module on the TabMWP task. . 134

7.16 The prompt constructed for the "Table Verbalizer" module on the TabMWP task. . 135

7.17 The prompt constructed for the "Program Generator" module on the TabMWP task. 135

7.18 The prompt constructed for the "Solution Generator" module on the TabMWP task. 136

7.19 An example of failure cases in ScienceQA from Chameleon (GPT-4) where *some modules perform inaccurately*. 137

7.20 An example of failure cases in TabMWP from Chameleon (GPT-4) where *some modules perform inaccurately and the generated programs are suboptimal*. 138

8.1 Definitions and proportions of seven mathematical reasoning categories in MathVista. 143

8.2 Key statistics of the MathVista benchmark. 147

8.3 Accuracy scores on the *testmini* subset of MathVista. Input: Q: question, I: image, I_c: image caption, I_t: OCR text detected in the image. 149

9.1 State-of-the-art performance scores of mathematical reasoning benchmarks in textual contexts (MATH and GSM8K) and visual contexts (MathVista). Statistics date: March 5, 2024. 157

ACKNOWLEDGMENTS

I would like to extend my deepest and most heartfelt thanks to my advisors and committee co-chairs, Prof. Kai-Wei Chang and Prof. Song-Chun Zhu. Prof. Chang has been exceptionally kind, supportive, and helpful. I am deeply appreciative of his professional guidance in my research, invaluable advice on fellowship applications, dedicated assistance in my job searches, and ongoing counsel on career development. Equally, I am profoundly grateful to Prof. Zhu for his visionary approach to research directions and his enduring dedication to academia. He is a brilliant philosopher, an inspiring leader, a supportive mentor, and a thoughtful elder. Both have been beacons, illuminating my path in academia and significantly molding my research.

I would like to express my appreciation to my committee members: Prof. Ying Nian Wu, Prof. Nanyun (Violet) Peng, and Prof. Guy Van den Broeck. Prof. Wu is like a friend, always offering steadfast support in both academic and personal matters. Prof. Peng is sincere and enthusiastic, providing meticulous guidance on academia and life. Prof. Van den Broeck has provided me with invaluable advice on my research, from which I will greatly benefit in my future career.

I am grateful for the opportunity to work with esteemed researchers Tong Sun and Jiuxiang Gu at Adobe Research; Peter Clark, Ashwin Kalyan, and Oyvind Tafjord at the Allen Institute for AI (AI2); and Jianfeng Gao, Michel Galley, Baolin Peng, Hao Cheng, and Chunyuan Li at Microsoft Research during my internships. I completed multiple projects during my time at AI2 and Microsoft Research. I truly appreciate the continued support from Peter Clark, Jianfeng Gao, and Michel Galley for my career development.

I would like to express my sincere gratitude to the numerous professors who have collaborated with me on projects and provided invaluable career advice throughout my academic journey, including Prof. Muhao Chen, Prof. Wenhu Chen, Prof. Noah Goodman, Prof. Hannaneh Hajishirzi, Prof. Heng Ji, Prof. Tianmin Shu, Prof. Yizhou Sun, Prof. Wei Wang, Prof. Sean Welleck, Prof. Wei Xu, Prof. Lin Yang, Prof. Zhou Yu, Prof. Jieyu Zhao, and Prof. James Zou.

I would like to thank all collaborators, especially those who contributed to my dissertation: Hritik Bansal, Jiaqi Chen, Yihe Deng, Matthew Finlayson, Peng Gao, Steven Gong, Jia-Chen Gu,

Ziniu Hu, Siyuan Huang, Shibiao Jiang, Jiacheng Liu, Xiao Liu, Yujie Lu, Swaroop Mishra, Liang Qiu, Tanmay Rajpurohit, Sheng Shen, Weiyan Shi, Robert Tang, Xiaoxuan Wang, Xueqing Wu, Tony Xia, Fan Yin, Wenhao Yu, Renrui Zhang, Yizhou Zhao, and Yanqiao Zhu.

Furthermore, I would like to express my appreciation to the members and friends of the NLP group, including Kareem Ahmed, Christina Chance, Amita Kamath, Harold Li, Zongyu Lin, Tao Meng, Masoud Monajatipoor, Anaelia Ovalle, Tanmay Parekh, Arjun Subramonian, Elaine Wan, Yiwei Wang, Di Wu, Cheng-Fu Yang, and Wade Yin, for a wonderful and stimulating experience you give me.

My thanks also extend to the VCLA group colleagues: Arjun Akula, Chao Xu, Yixin Chen, Mark Edmonds, Lifeng Fan, Ruiqi Gao, Xiaofeng Gao, Muzhi Han, Mitchell Hill, Yining Hong, Baoxiong Jia, Ziyuan Jiao, Deqian Kong, Qing Li, Hangxin Liu, Tengyu Liu, Qian Long, Xiaojian Ma, Jonathan Mitchell, Erik Nijkamp, Shuwen Qiu, Feng Shi, Shu Wang, Weiqi Wang, Sirui Xie, Xu Xie, Dehong Xu, Yifei Xu, Luyao Yuan, Chi Zhang, Victor Zhang, Zeyu Zhang, Tianyang Zhao, Zilong Zheng, Yaxuan Zhu, and Yixin Zhu. I have greatly benefited from their supportive and insightful chats and discussions.

I would like to thank the Plus group peers, especially for a vibrant, happy, and enjoyable time, including Lucas Bandarkar, Zi-Yi Dou, Mohsen Fayyaz, I-Hung Hsu, Wenbo Hu, Po-Nien Kung, Sidi Lu, Rebacca Pattichis, Haoyi Qiu, Jiao Sun, Ashima Suvarna, Rohan Wadhawan, Wenxuan Wang, Te-Lin Wu, Songyan Zhao, Chujie Zheng, and Yu Zhou.

Finally, my heartfelt thanks to my parents and my soul partner for their unwavering support and love.

VITA

2019–2024	Graduate Student Researcher, UCLA.
2021–2023	Teaching Assistant / Associate, UCLA.
2015–2018	M.E. in Computer Science, Tsinghua University.
2011–2015	B.S. in Electrical Engineering, Beijing Institute of Technology.

PUBLICATIONS

(* indicates Equal Contribution)

X. Wang*, Z. Hu*, **P. Lu***, Y. Zhu*, J. Zhang, S. Subramaniam, A.R. Loomba, S. Zhang, Y. Sun, W. Wang. *SciBench: Evaluating college-level scientific problem-solving abilities of large language models*. International Conference on Machine Learning (ICML), 2024.

P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, J. Gao. *MathVista: Evaluating mathematical reasoning of foundation models in visual contexts*. International Conference on Learning Representations (ICLR), 2024.

X. Liu, Z. Wu, **P. Lu**, K.-W. Chang, Y. Feng. *Are LLMs capable of data-based statistical and causal reasoning? benchmarking advanced quantitative reasoning with data*. Findings of Annual Meeting of the Association for Computational Linguistics (ACL), 2024.

W. Chen, M. Yin, M. Ku, **P. Lu**, Y. Wan, X. Ma, J. Xu, X. Wang, T. Xia. *TheoremQA: A theorem-driven question answering dataset*. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023.

P. Lu, B. Peng, H. Cheng, M. Galley, K.-W. Chang, Y.N. Wu, S.-C. Zhu, J. Gao. *Chameleon: Plug-and-play compositional reasoning with large language models*. Advances in Neural Information Processing Systems (NeurIPS), 2023.

P. Lu, L. Qiu, W. Yu, S. Welleck, K.-W. Chang. *A survey of deep learning for mathematical reasoning*. Annual Meeting of the Association for Computational Linguistics (ACL), 2023.

P. Lu, L. Qiu, K.-W. Chang, Y.N. Wu, S.-C. Zhu, T. Rajpurohit, P. Clark, A. Kalyan. *Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning*. International Conference on Learning Representations (ICLR), 2023.

S. Mishra*, M. Finlayson*, **P. Lu**, L. Tang, S. Welleck, C. Baral, T. Rajpurohit, O. Tafjord, A. Sabharwal, P. Clark, A. Kalyan. *Lila: A unified benchmark for mathematical reasoning*. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022.

J. Chen, T. Li, J. Qin, **P. Lu**, L. Lin, C. Chen, X. Liang. *UniGeo: Unifying geometry logical reasoning via reformulating mathematical expression*. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022.

P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, A. Kalyan. *Learn to explain: Multimodal reasoning via thought chains for science question answering*. Advances in Neural Information Processing Systems (NeurIPS), 2022.

L. Qiu, Y. Zhao, Y. Liang, **P. Lu**, W. Shi, Z. Yu, S.-C. Zhu. *Towards socially intelligent agents with mental state transition and human utility*. Special Interest Group on Discourse and Dialogue (SIGDIAL), 2022.

Y. Zhao, L. Qiu, **P. Lu**, F. Shi, T. Han, S.-C. Zhu. *Learning from the tangram to solve mini visual tasks*. Association for the Advancement of Artificial Intelligence (AAAI), 2022.

L. Qiu, Y. Zhao, J. Li, **P. Lu**, B. Peng, J. Gao, S.-C. Zhu. *ValueNet: A new dataset for human value driven dialogue system*. Association for the Advancement of Artificial Intelligence (AAAI), 2022.

P. Lu, L. Qiu, J. Chen, T. Xia, Y. Zhao, W. Zhang, Z. Yu, X. Liang, S.-C. Zhu. *IconQA: A new benchmark for abstract diagram understanding and visual language reasoning*. Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, 2021.

L. Qiu, Y. Liang, Y. Zhao, **P. Lu**, B. Peng, Z. Yu, Y.N. Wu, S.-C. Zhu. *SocAoG: incremental graph parsing for social relation inference in dialogues*. Annual Meeting of the Association for Computational Linguistics (ACL), 2021.

P. Lu*, R. Gong*, S. Jiang*, L. Qiu, S. Huang, X. Liang, S.-C. Zhu. *Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning*. Annual Meeting of the Association for Computational Linguistics (ACL), 2021.

CHAPTER 1

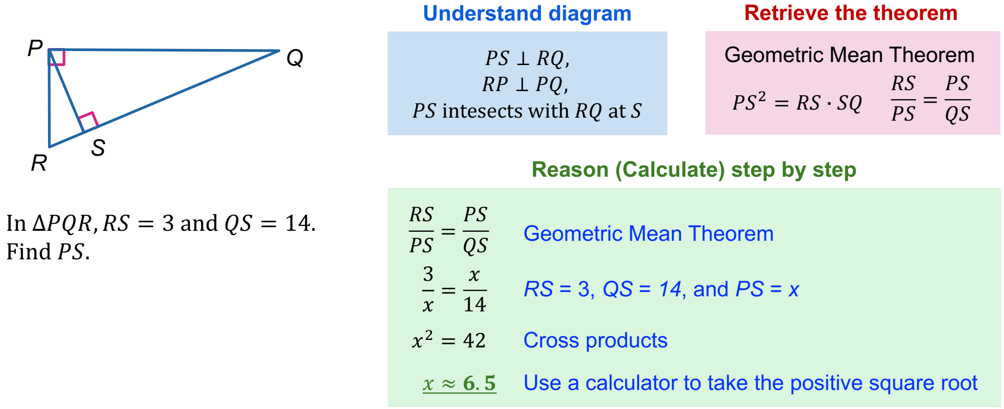
Introduction

1.1 Motivation

Mathematical reasoning is a fundamental aspect of human intelligence that enables us to comprehend and make decisions based on numerical data and language. It is applicable in various fields, including science, engineering, finance, and everyday life, and encompasses a range of abilities, from basic skills such as pattern recognition and numerical operations to more advanced skills like problem-solving, logical reasoning, and abstract thinking. The development of artificial intelligence (AI) systems capable of solving math problems and proving theorems has been a long-standing focus of research in machine learning and natural language processing (NLP), dating back to the 1960s [Fei63, Bob64].

As deep learning continues to revolutionize NLP tasks such as question answering and machine translation [SVL14, KT19], it has also made significant strides in mathematical reasoning [WLS17, YD19, GGB20, WWS22b]. Recently, large language models (LLMs) such as GPT-4 [Ope23a] and LLaMA 3 [TLI23] have demonstrated impressive mathematical reasoning capabilities, achieving comparable human-level performance on benchmarks like GSM8K [CKB21].

However, current works face several challenges. First, the focus has primarily been on text-only domains [CKB21, HBK21], while real-world math and science problems often involve multimodal contexts such as images, diagrams, and tables. Second, LLMs are approaching performance saturation on some text-based benchmarks, leaving limited room for improvement [Ant24, ZWL23]. Third, there remains a significant performance gap on more complex, real-world challenges that require retrieving and applying domain knowledge, conducting multi-step reasoning, and utilizing domain-specific tools [LBX24].



In ΔPQR, $RS = 3$ and $QS = 14$. Find PS.

Understand diagram

$PS \perp RQ$,
$RP \perp PQ$,
PS intesects with RQ at S

Retrieve the theorem

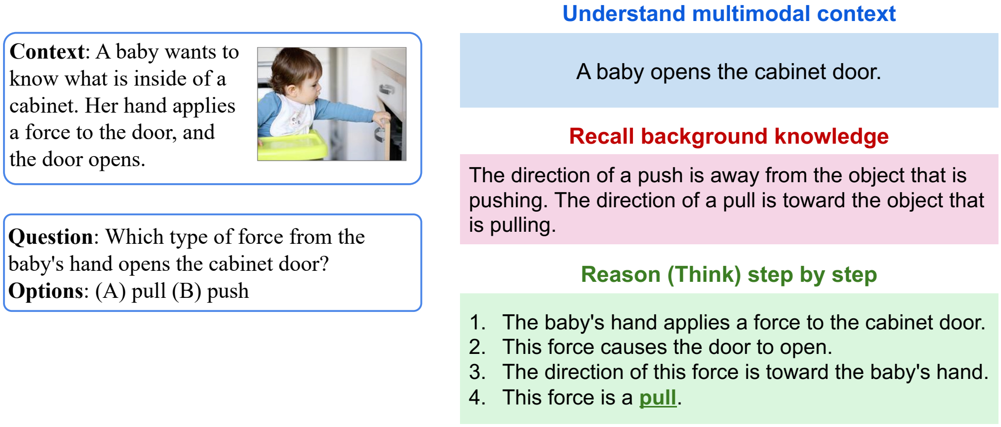
Geometric Mean Theorem
$PS^2 = RS \cdot SQ$ $\quad \frac{RS}{PS} = \frac{PS}{QS}$

Reason (Calculate) step by step

$\frac{RS}{PS} = \frac{PS}{QS}$ Geometric Mean Theorem
$\frac{3}{x} = \frac{x}{14}$ $RS = 3$, $QS = 14$, and $PS = x$
$x^2 = 42$ Cross products
$\underline{x \approx \mathbf{6.5}}$ Use a calculator to take the positive square root

Figure 1.1: A geometry problem requiring diagram understanding, theorem selection, and step-by-step calculation.

To illustrate the challenges faced by current LLMs, consider three example problems involving geometry diagrams, scientific scenarios, and tabular contexts. Solving the geometry problem in Figure 1.1 requires understanding the diagram, grounding symbols and attributes, and applying appropriate theorems like the Geometric Mean Theorem. While the Pythagorean theorem might initially seem applicable, it would lead to a complex system of equations. Instead, the Geometric Mean Theorem allows for a more straightforward solution by setting up a single equation and calculating step by step.



Context: A baby wants to know what is inside of a cabinet. Her hand applies a force to the door, and the door opens.

Question: Which type of force from the baby's hand opens the cabinet door?
Options: (A) pull (B) push

Understand multimodal context

A baby opens the cabinet door.

Recall background knowledge

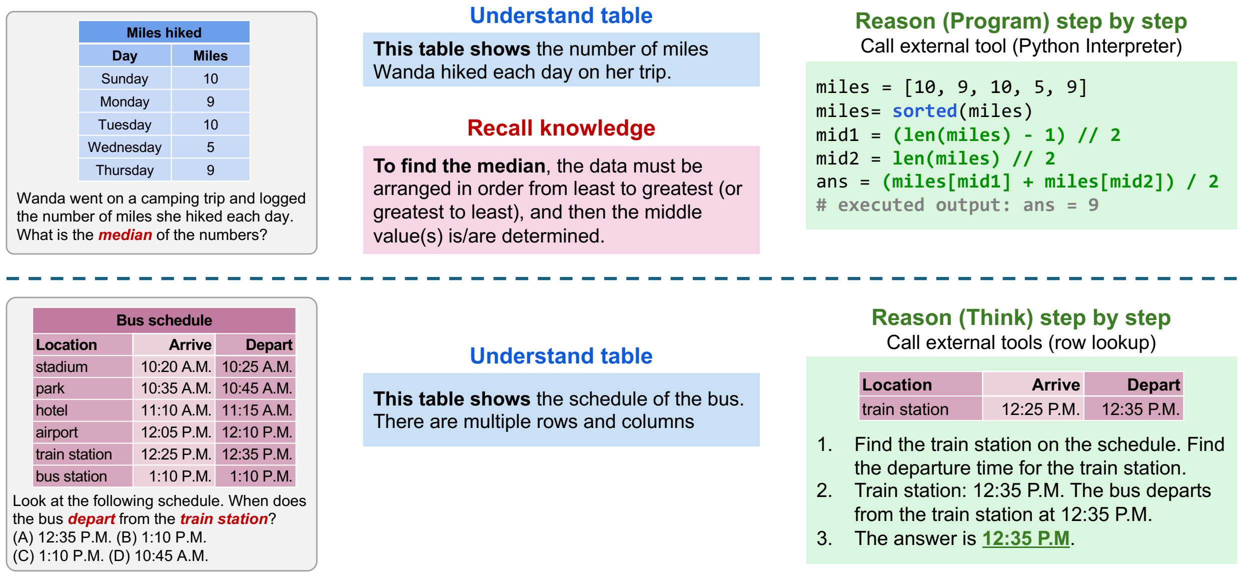
The direction of a push is away from the object that is pushing. The direction of a pull is toward the object that is pulling.

Reason (Think) step by step

1. The baby's hand applies a force to the cabinet door.
2. This force causes the door to open.
3. The direction of this force is toward the baby's hand.
4. This force is a **pull**.

Figure 1.2: A science question requiring multimodal understanding, knowledge retrieval, and multi-hop reasoning.

Figure 1.2 presents a science question that requires understanding the multimodal context of a baby opening a cabinet door, recalling relevant background knowledge, and providing step-by-step reasoning based on the context and knowledge to arrive at the correct answer. Answering such questions involves multimodal content understanding, external knowledge extraction, and explicit multi-hop reasoning.


Understand table
Recall knowledge
Reason (Program) step by step
Call external tool (Python Interpreter)
Understand table
Reason (Think) step by step
Call external tools (row lookup)

Miles hiked	
Day	**Miles**
Sunday	10
Monday	9
Tuesday	10
Wednesday	5
Thursday	9

Wanda went on a camping trip and logged the number of miles she hiked each day. What is the ***median*** of the numbers?

This table shows the number of miles Wanda hiked each day on her trip.

To find the median, the data must be arranged in order from least to greatest (or greatest to least), and then the middle value(s) is/are determined.

```
miles = [10, 9, 10, 5, 9]
miles= sorted(miles)
mid1 = (len(miles) - 1) // 2
mid2 = len(miles) // 2
ans = (miles[mid1] + miles[mid2]) / 2
# executed output: ans = 9
```

Bus schedule		
Location	**Arrive**	**Depart**
stadium	10:20 A.M.	10:25 A.M.
park	10:35 A.M.	10:45 A.M.
hotel	11:10 A.M.	11:15 A.M.
airport	12:05 P.M.	12:10 P.M.
train station	12:25 P.M.	12:35 P.M.
bus station	1:10 P.M.	1:10 P.M.

Look at the following schedule. When does the bus ***depart*** from the ***train station***?
(A) 12:35 P.M. (B) 1:10 P.M.
(C) 1:10 P.M. (D) 10:45 A.M.

This table shows the schedule of the bus. There are multiple rows and columns

Location	Arrive	Depart
train station	12:25 P.M.	12:35 P.M.

1. Find the train station on the schedule. Find the departure time for the train station.
2. Train station: 12:35 P.M. The bus departs from the train station at 12:35 P.M.
3. The answer is **12:35 P.M**.

Figure 1.3: Math word problems in tabular context requiring knowledge retrieval and tool utilization.

Figure 1.3 shows two math word problems in tabular context. The first question, asking for the median of numbers, requires recalling how to calculate the median and potentially utilizing a Python interpreter for a precise result. The second question, involving a larger table, could be simplified by using a row lookup tool to locate relevant information.

Solving these problems requires capabilities beyond just processing text, such as understanding visual information, retrieving relevant knowledge, and performing rigorous logical reasoning, potentially with the aid of external tools. Current LLMs, while powerful, are not well-equipped to handle such multimodal, knowledge-intensive problems requiring complex reasoning.

To address these challenges, this dissertation first proposes **multimodal, knowledge-intensive benchmarks** to evaluate the capabilities of LLMs on real-world math and science problems. These benchmarks are designed to be (1) *multimodal*, featuring not just text but also natural images, geometry diagrams, abstract scenes, and scientific figures; (2) *knowledge-intensive*, requiring retrieval and application of domain knowledge, online resources, textbooks, theorems, and rules; and (3) requiring *rigorous multistep reasoning*, often needing neuro-symbolic reasoning, tool augmentation, and verification to ensure precise predictions.

The dissertation then explores **pre-trained vision-language models (VLMs)** to enhance mathematical reasoning through pretraining and hierarchical image parsing layouts. Furthermore, it proposes **tool-augmented and retrieval-augmented algorithms** that significantly enhance LLMs' mathematical reasoning in real scenarios. The key insights are that (1) integrating formal languages and symbolic solvers can strengthen LLMs for complex reasoning, such as in geometry, (2) LLMs can learn to dynamically select relevant demonstrations to improve in-context learning performance, and (3) LLMs can effectively utilize external tools and resources.

The ultimate goal is to develop LLMs and their visual variants VLMs that can more robustly and flexibly apply mathematical reasoning to solve real-world problems across diverse domains by leveraging multimodal understanding, retrieving and applying relevant knowledge, composing necessary tools, and generating correct solutions through interpretable multi-step reasoning. This dissertation takes important steps towards that goal through new benchmarks, models, and algorithms while also revealing key challenges and future opportunities in this space.

1.2 Dissertation Outline

My ultimate research goal is to **build machines that can reason and collaborate with humans for the common good**, an ambition that lies at the intersection of AI, humans, and science. Specifically, I aim to **develop AI systems that achieve expert-level problem-solving and assist humans in scientific discovery**.

Following these central topics, the dissertation is organized into four parts as follows: PART I MULTIMODAL AND KNOWLEDGE-INTENSIVE BENCHMARKS, PART II PRE-TRAINED VISION-LANGUAGE MODELS, PART III RETRIEVAL AND TOOL-AUGMENTED ALGORITHMS, and PART IV SUMMARY AND FUTURE DIRECTIONS.

PART I introduces novel mathematical reasoning benchmarks in multimodal and knowledge-intensive scenarios. Chapter 2 presents three new datasets respectively: Geometry3K [LGJ21], IconQA [LQC21], and TabMWP [LQC23], which are designed to evaluate AI models' performance in rigorous math problems across diverse contexts such as geometry diagrams, abstract scenes, and tabular data. These benchmarks provide a comprehensive testbed for assessing the capabilities of AI systems in mathematical reasoning tasks that require understanding and integrating information from multiple modalities. Chapter 3 introduces ScienceQA [LMX22], a benchmark that assesses AI models' performance in knowledge-intensive scientific reasoning tasks requiring multimodal understanding and external knowledge application. ScienceQA serves as a valuable resource for evaluating the ability of AI systems to reason about complex scientific concepts and leverage domain-specific knowledge.

PART II explores the development of robust foundation models for mathematical reasoning in abstract diagrams. Chapter 4 proposes Patch-Trm [LQC21], a novel patch-based cross-modal Transformer model that enhances visual mathematical reasoning capabilities by learning hierarchical relationships between image patches and text. Patch-Trm demonstrates the potential of specialized architectures in improving the performance of AI systems on visual reasoning tasks, particularly in the context of abstract diagrams.

PART III focuses on the design of efficient algorithms to enhance reasoning abilities, through tool-augmented and retrieval-augmented approaches. Chapter 5 introduces Inter-GPS [LGJ21], a neuro-symbolic solver for geometry problems that demonstrates human-level performance. Inter-GPS showcases the effectiveness of combining neural networks with symbolic reasoning techniques to tackle complex geometric reasoning tasks. Chapter 6 presents PromptPG [LQC23], a novel approach that employs reinforcement learning for dynamic in-context example selection to improve the few-shot learning capabilities of large language models. PromptPG highlights the potential of adaptive prompting strategies in enhancing the performance of language models on reasoning tasks with limited training data. Chapter 7 introduces Chameleon [LPC23], a tool-augmented reasoning framework that enhances the reasoning capabilities of language models by integrating external computational tools. Chameleon demonstrates the benefits of augmenting language models with specialized tools and knowledge sources to tackle complex reasoning problems.

PART IV summarizes recent progress in mathematical reasoning and discusses future directions. Chapter 8 highlights the latest advances in visual mathematical reasoning [LBX24], addressing current challenges and potential future directions, such as performance saturation, bridging performance gaps, and strategies for self-improvement. This chapter provides an overview of the state-of-the-art in visual mathematical reasoning and identifies key areas for future research and development. Chapter 9 concludes the dissertation by discussing future work and the broader impact of the research, emphasizing the potential of AI systems to achieve expert-level problem-solving and assist humans in scientific discovery. The conclusion underscores the significance of the presented research in advancing the field of AI and its potential to transform various domains through enhanced reasoning capabilities.

By curating novel benchmarks, developing robust foundation models, and designing efficient algorithms, this dissertation aims to push the boundaries of AI's capabilities in complex reasoning tasks and facilitate scientific discoveries. The research presented here contributes to the overarching goal of building machines that can reason and collaborate with humans for the common good, marking a significant step towards realizing general artificial intelligence.

Part I

Multimodal and Knowledge-Intensive Benchmarks

CHAPTER 2

Multimodal Mathematical Reasoning

2.1 Geometry Problem Solving

2.1.1 Introduction

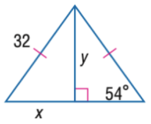
Automated geometry problem solving (GPS) is also a long-standing mathematical reasoning task [GHL60, Wen86]. As shown in Figure 1.1, a geometry problem consists of a textual description and a diagram. The multimodal inputs describe the entities, attributes, and relationships of geometric elements, and the goal is to find the numeric solution to an unknown variable.

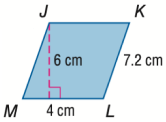
Several datasets for geometry have been released in recent years, including GEOS [SHF15], GEOS++ [SDX17], GeoShader [AGM17] and GEOS-OS [SX17] datasets. However, these datasets are relatively small in scale and contain limited problem types. For example, there are only 102 shaded area problems in GeoShader and 186 problems in GEOS. While GEOS++ and GEOS-OS contain more data of 1,406 and 2,235 problems, respectively, they have not been publicly available yet. Instead, our Geometry3K dataset features 3,002 SAT-style problems collected from two high-school textbooks that cover diverse graph and goal types. Besides, each problem in Geometry3K is annotated with dense descriptions in formal language (defined in Section 5.3), which makes it particularly suited for symbolic reasoning and interpretable problem solving.

2.1.2 Dataset Collection

Most existing datasets for geometry problem solving are relatively small, contain limited problem types, or not publicly available. For instance, the GEOS dataset [SHF15] only contains 186 SAT problems. Although there are 1,406 problems in GEOS++ [SDX17], this dataset has not been

Problem Text	**Diagram**	**Choices**	**Text Literals**	**Diagram Literals**

Find y. Round to the nearest tenth.		A. 18.8
		B. 23.2
		C. 25.9
		D. 44.0
		Answer: C
			`Find(y)`	`Equals(LengthOf(Line(A,B)),32)`
				`Equals(LengthOf(Line(B,D)),y)`
				`Equals(MeasureOf(Angle(A,C,B)),54)`
				`Equals(LengthOf(Line(A,D)),x)`
				`PointLiesOnLine(D,Line(A,C))`
				`Perpendicular(Line(B,D),Line(C,D))`
				`Equals(LengthOf(Line(A,B)),LengthOf(Line(B,C)))`

Find the perimeter of $\parallelogram$ JKLM.		A. 11.2
		B. 22.4
		C. 24
		D. 44.8
		Answer: B
			`Find(PerimeterOf(Parallelogram(J,K,L,M)))`	`Equals(LengthOf(Line(L,K)),7.2)`
				`Equals(LengthOf(Line(M,L)),4)`
				`Equals(LengthOf(Line(E,J)),6)`
				`PointLiesOnLine(E,Line(M,L))`
				`Perpendicular(Line(J,E),Line(E,L))`

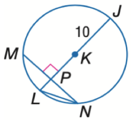
In $\odot$ K, MN = 16 and m $\widehat$ MN = 98. Find the measure of LN. Round to the nearest hundredth.		A. 6.93
		B. 7.50
		C. 8.94
		D. 10.00
		Answer: C
			`Circle(K)`
			`Equals(LengthOf(Line(M,N)),16)`
			`Equals(MeasureOf(Arc(M,N)),98)`
			`Find(LengthOf(Line(L,N)))`
				`Equals(LengthOf(Line(J,K)),10)`
				`Perpendicular(Line(P,K),Line(M,P))`
				`PointLiesOnLine(P,Line(M,N))`
				`PointLiesOnLine(P,Line(L,J))`
				`PointLiesOnLine(P,Line(L,K))`
				`PointLiesOnLine(K,Line(P,J))`
				`PointLiesOnLine(K,Line(L,J))`
				`PointLiesOnCircle(M,Circle(K))`
				`PointLiesOnCircle(J,Circle(K))`
				`PointLiesOnCircle(N,Circle(K))`
				`PointLiesOnCircle(L,Circle(K))`

Figure 2.1: More data examples in the Geometry3K dataset.

released to the public yet. Therefore, we build a new large-scale geometry problem benchmark, called Geometry3K. The data is collected from two popular textbooks for high school students across grades 6-12 by two online digital libraries (McGraw-Hill[1], Geometryonline[2]). Groups of well-trained annotators with undergraduate degrees manually collect each problem with its problem text, geometry diagram, four candidate choices, and correct answer. Each problem text is annotated in the format of LaTeX. In order to evaluate the fine-grained performance of geometry solvers, we label each problem data with the corresponding problem goal and geometry shapes.

Unlike existing datasets that only collect the problem text and diagrams, we further annotate each data in Geometry3K with dense formal language descriptions that bridge the semantic gap between the textual and visual contents as well as benefit the symbolic problem solver. The annotated formal language is used to train and evaluate our proposed problem parsers. Data examples are illustrated in Figure 2.1.

---

[1] `https://www.mheducation.com/`

[2] `www.geometryonline.com`

	Total	**Train**	**Val**	**Test**
Questions	3,002	2,101	300	601
Sentences	4,284	2,993	410	881
Words	30,146	20,882	2,995	6,269
Literals (Text)	6,293	4,357	624	1,312
Literals (Diagram)	27,213	19,843	2,377	4,993

Table 2.1: Basic statistics of our Geometry3K dataset.

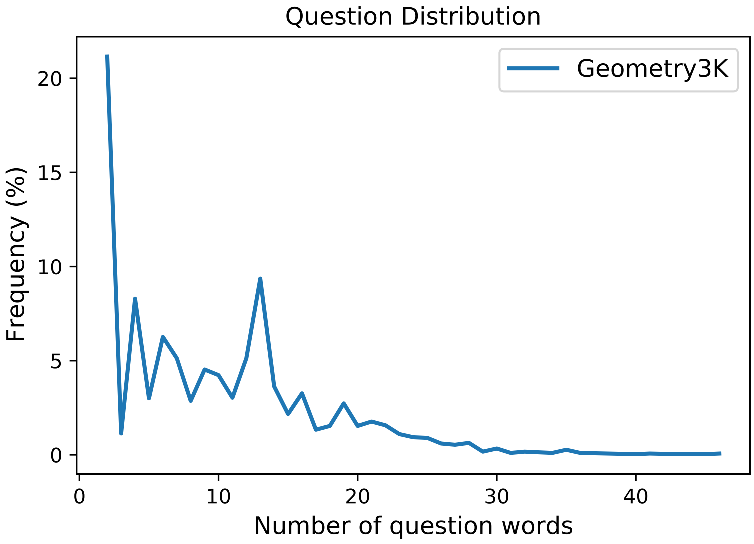


Figure 2.2: Question length distribution of Geometry3K.

2.1.3 Dataset Statistics

The Geometry3K dataset consists of 3,002 problems and is divided into the train, validation, and test sets with the ratio of 0.7:0.1:0.2, as shown in Table 2.1. Figure 2.2 illustrates the question distribution by the number of sentence words. The long tail in the distribution requires the geometry solvers to understand the rich semantics in the textual content.

There are 6,293 literals for the problem text and 27,213 literals for the diagrams in Geometry3K, respectively. We list the most and least frequent predicates with a frequency greater than 5 in Table 2.2. It is shown that the predicates for the problem text are more evenly distributed than those for diagrams. This is mainly because the problem text describes diverse geometric shapes, attributes, and relations while diagrams display the basic properties of points, lines, and arcs.

Predicates (Text)	**%**	**Predicates (Diagram)**	**%**
`Find`	19.00	`Line`	30.89
`Line`	14.49	`PointLiesOnLine`	16.66
`Equals`	11.83	`Equals`	15.17
`LengthOf`	9.53	`MeasureOf`	10.46
`MeasureOf`	8.97	`LengthOf`	8.69
……		……	
`CircumscribedTo`	0.05	`Triangle`	0.03
`SumOf`	0.04	`Quadrilateral`	0.02
`HeightOf`	0.04	`Kite`	0.01
`BaseOf`	0.04	`HeightOf`	0.01
`IsHypotenuseOf`	0.04	`Square`	0.01

Table 2.2: Most and least frequent predicates in the formal descriptions of the Geometry3K dataset (with a frequency greater than 5).

Dataset	**#qa**	**#word**	**#shape**	**#goal**	**#var**	**grade**	**operator type**
GeoShader [AGM17]	102	/	4	1	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
GEOS [SHF15]	186	4,343	4	3	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
GEOS++ [SDX17]	1,406	/	4	3	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
GEOS-OS [SX17]	2,235	/	4	3	1	6-10	$\{+, -, \times, \div, \square^2, \sqrt{\square}\}$
Geometry3K (ours)	3,002	36,736	6	4	3	6-12	$\{+, -, \times, \div, \square^2, \sqrt{\square}, \sin, \cos, \tan\}$

Table 2.3: Comparison of our Geometry3K dataset with existing datasets.

2.1.4 Comparisons with Existing Datasets

To the best of our knowledge, it is one of the largest geometry problem datasets. We summarize the Geometry3K dataset's main statistics and a comparison of existing datasets in Table 2.3. In addition to four elementary shapes (lines, triangles, regular quadrilaterals, and circles) mentioned in that GEOS dataset, Geometry3K contains irregular quadrilaterals and other polygons. Besides, in Geometry3K, there are more unknown variables and operator types that may require equation solving to find the goal of the problem. Note that 80.5% of problems are solvable without the associated diagram in the GEOS dataset. By contrast, less than 1% of the problems in our Geometry3K dataset could be solved when the problem diagram is not provided. In general, the statistics and comparisons above show Geometry3K is challenging for geometry problem solvers.

2.1.5 Human Performance

We pushed the test-split data of the dataset to the crowdsourcing platform, Amazon Mechanical Turk[3]. Each eligible annotator must have obtained a high school or higher degree and was asked to answer 10 problems within 25 minutes. To ensure annotators solved the problems to the best of their ability, they were further asked to spend at least 7 minutes on the problem set and 10 seconds on each problem. We filtered out annotators who did not satisfy these requirements. We also asked dozens of graduates majoring in science or engineering to answer these problems to evaluate human experts' performance. Table 5.2 shows the human performance. Compared to random guessing with an accuracy of 25%, humans achieved an overall accuracy of 56.9%, and human experts achieved a good performance of 90.9%.

2.2 Math Word Problems in Abstract Scenes

2.2.1 Introduction

In recent years, we have witnessed exciting advancements in the field of visual question answering (VQA). This area of research aims to develop systems capable of responding to natural language questions based on visual information. There have been efforts to develop datasets for the visual question answering (VQA) task since the first large-scale benchmark was introduced in [AAL15]. Early released datasets [GKS17, KZG17, SNS19, WLS20] contain natural images and related questions, where understanding the visual and textual contents is essential for question answering. Some recent datasets introduce questions that involve more diverse visual scenes or require external knowledge to answer, which leads to more complex visual and semantic reasoning for question answering. For example, CLEVR [JHM17] is a synthetic dataset that serves as a diagnostic test for a range of visual reasoning abilities over combinations of three object shapes. However, these datasets are limited to the natural image domain and pay little attention to abstract diagrams, which also have informative semantics and wide applications.

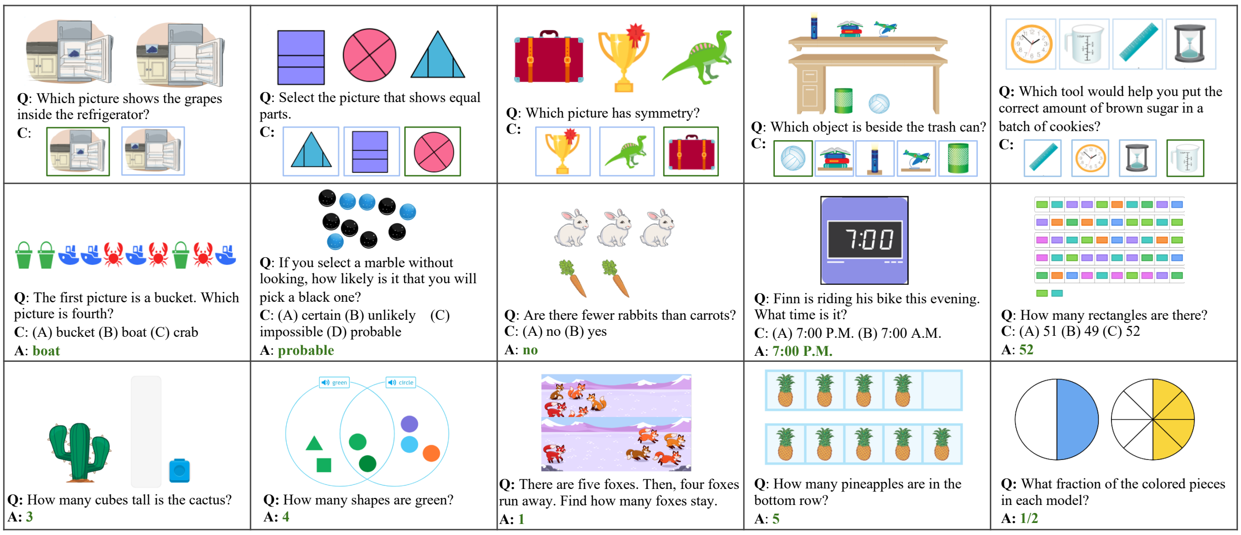
---

[3] `https://www.mturk.com/`

To address the need for vision-and-language reasoning for diagrams, several abstract diagram QA datasets have been developed. For example, abstract VQA [AAL15, ZGS16] considers the task of answering questions on abstract scenes. Similarly, NLVR [SLY17], FigureQA [KMA17], and DVQA [KPC18] feature diagrams that are generated with several figure types or question templates. However, either diagrams or questions in these datasets are generated from limited templates, leading to the existence of unintended visual or linguistic shortcuts for question answering. Some more works have proposed datasets of middle school math or science problems in more practical and complex scenarios [SHF15, KSS17, SDX17, SDM18, LGJ21]. A central limitation of the subject QA datasets is that they require complex domain-specific knowledge, which makes disentangling visual reasoning and domain knowledge difficult. Herein, we address these limitations by introducing the IconQA dataset, where only elementary commonsense is required. Through IconQA, we aim to provide a new benchmark for abstract scene understanding and learning different visual reasoning skills in *real-world* scenarios.

To address these shortcomings, we release Icon Question Answering (IconQA), a large-scale dataset that contains 107,439 QA pairs and covers three different sub-tasks: *multiple-image-choice*, *multiple-text-choice* and *filling-in-the-blank*. A typical IconQA problem is provided with an icon image and a question, and the answer is in the form of either a short piece of text or a choice from multiple visual or textual choices. Correctly answering IconQA questions needs diverse human intelligence skills. As examples in Figure 2.3 show, IconQA poses new challenges for abstract diagram understanding like recognizing objects and identifying attributes. Besides, it is critical to develop diverse cognitive reasoning skills, including counting objects, comparing attributes, performing arithmetic operations, making logical inferences, completing spatial reasoning, or leveraging external commonsense to answer IconQA questions.

The IconQA dataset provides diverse questions that require abstract diagram recognition, comprehensive visual reasoning skills, and basic commonsense knowledge. IconQA consists of 107,439 questions split across three different sub-tasks. To the best of our knowledge, IconQA is the largest VQA dataset that focuses on real-world problems with icon images while involving multiple human intelligence reasoning abilities (see Table 2.8).



Q: Which picture shows the grapes inside the refrigerator?
C:

Q: Select the picture that shows equal parts.
C:

Q: Which picture has symmetry?
C:

Q: Which object is beside the trash can?
C:

Q: Which tool would help you put the correct amount of brown sugar in a batch of cookies?
C:

Q: The first picture is a bucket. Which picture is fourth?
C: (A) bucket (B) boat (C) crab
A: **boat**

Q: If you select a marble without looking, how likely is it that you will pick a black one?
C: (A) certain (B) unlikely (C) impossible (D) probable
A: **probable**

Q: Are there fewer rabbits than carrots?
C: (A) no (B) yes
A: **no**

Q: Finn is riding his bike this evening. What time is it?
C: (A) 7:00 P.M. (B) 7:00 A.M.
A: **7:00 P.M.**

Q: How many rectangles are there?
C: (A) 51 (B) 49 (C) 52
A: **52**

Q: How many cubes tall is the cactus?
A: **3**

Q: How many shapes are green?
A: **4**

Q: There are five foxes. Then, four foxes run away. Find how many foxes stay.
A: **1**

Q: How many pineapples are in the bottom row?
A: **5**

Q: What fraction of the colored pieces in each model?
A: **1/2**

Figure 2.3: Examples from the IconQA dataset. Top: *multi-image-choice* sub-task. Middle: *multi--text-choice* sub-task. Bottom: *filling-in-the-blank* sub-task.

2.2.2 Dataset Collection

We aim to collect icon-based question answering pairs that involve multiple reasoning skills, such as visual reasoning and commonsense reasoning. To construct the IconQA dataset, which stems from real-world math word problems, we search for open-source math textbooks with rich icon images and diverse topics. Of those, we choose *IXL Math Learning* which compiles popular textbooks aligned to California Common Core Content Standards[4]. We ask well-trained crowd workers to collect problems that cover content from pre-K to third grade, as these problems usually contain abstract images and involve little to none complex domain knowledge. With the driven interest of visual reasoning over abstract images, we filter out the questions that do not accompany icon images or only have images in black and white. Redundant or repetitive data instances are also removed. Question choices are randomly shuffled to ensure a balanced answer distribution.

Question skill categories. The questions we collected contain meta-information including question topics, chapter names, image names, and so on. After extensive data exploration by well-informed individuals, we designed a set of rules that map each question to 1-3 of the 13 categories based on trigger words in metadata. The rules for trigger words are listed in Table 2.4.

---

[4] `https://www.ixl.com/standards/california/math`

Skill types	Trigger words in metadata
Geometry	name the shape, shapes of, classify shapes, solid, corners, faces, edges, vertices, sides, dimensional, rectangle, circle, triangle, square, rhombus, sphere, cylinder, cone, cubes, hexagon, perimeter, area, curved, open and close, flip turn, symmetry
Counting	count, tally, a group, ordinal number, area, even or odd, place value, represent numbers, comparing review, equal sides, square corners, one more, one less, fewer, enough, more.
Comparing	compare, comparing, more, less, fewer, enough, wide and narrow, light and heavy, long and short, tall and short, match analog and digital
Spatial	top, above, below, beside, next to, inside and outside, left
Scene	problems with pictures, beside, above, inside and outside, wide and narrow, objects
Pattern	the next, comes next, ordinal number, different
Time	clock, am or pm, elapsed time, times
Fraction	equal parts, halves, thirds, fourths, fraction
Estimation	estimate, measure
Algebra	count to fill, skip count, tally, even or odd, tens and ones, thousands, of ten, elapsed time, perimeter, area, divide
Measurement	measure
Commonsense	light and heavy, compare size, holds more or less, am or pm, times of, tool
Probability	likely

Table 2.4: Trigger words in metadata used to categorize skills in the IconQA dataset.

2.2.3 Data Analysis

Finally, we collect 107,439 IconQA data instances, where each data point contains a colored icon image, a natural language question, optional image or text choices, as well as a correct answer. The IconQA dataset consists of 107,439 questions and is divided into train, validation, and test splits with a ratio of 6:2:2, as shown in Table 2.5. The dataset consists of three sub-tasks: *multi-image-choice*, *multi-text-choice*, and *filling-in-the-blank*. The *multi-image-choice* sub-task is defined as choosing the correct image from a list of image candidates based on a given diagram and its corresponding question. Similarly, the *multi-text-choice* sub-task is defined as a multiple choice question with 2-5 text choices and an abstract diagram. The *filling-in-the-blank* sub-task is similar to the common VQA task, requiring a brief text answer for each question, except in IconQA, the images are icon images instead of natural images.

Questions. Figure 2.4 illustrates the distribution of question lengths of each sub-task in the IconQA dataset. For simplicity, all questions longer than 35 words are counted as having 35 words. Questions in the *multi-text-choice* sub-task distribute more evenly, while for *multi-img-*

Tasks	**All**	**Train**	**Val**	**Test**
Multi-image-choice	57,672	34,603	11,535	11,535
Multi-text-choice	31,578	18,946	6,316	6,316
Filling-in-the-blank	18,189	10,913	3,638	3,638
All	107,439	64,462	21,489	21,489

Table 2.5: Statistics for the IconQA dataset.

choice, there is a long-tail distribution due to the complexity of textual scenarios. We find that some icon objects are frequently mentioned in the questions.

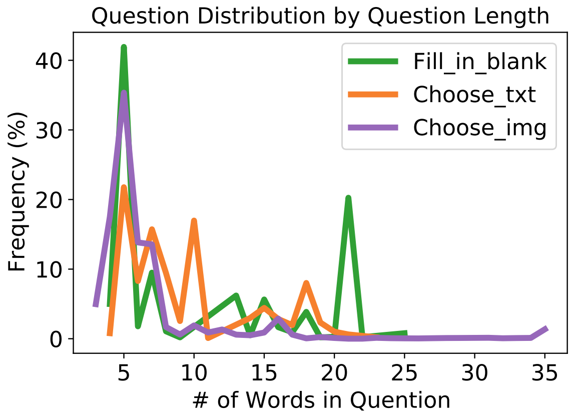


Figure 2.4: Question statistics based on number of words in IconQA.

In Figure 2.5, the frequencies of the 40 most frequently mentioned icons are shown. These icon entities cover different daily-life objects such as animals, plants, shapes, food, etc. We cluster question sentences into different types based on frequent trigram prefixes starting the sentences.

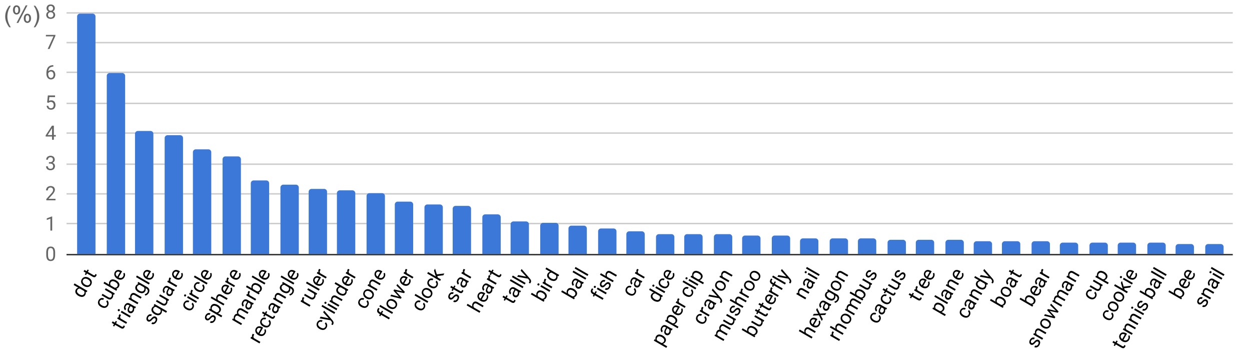


Figure 2.5: Top 40 icons mentioned in the IconQA question texts and their appearance percentage. These icons cover various types of real-world objects.

The distribution of questions is visualized in Figure 2.6. Importantly, the diversity in the ques-

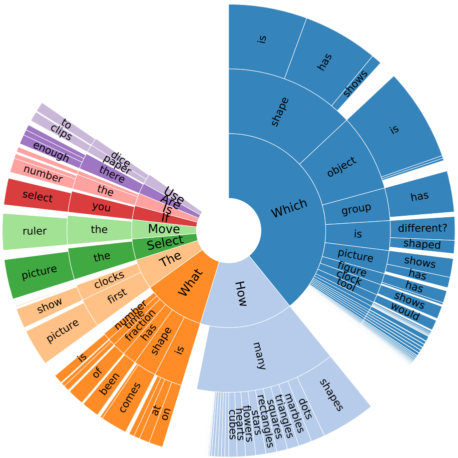


Figure 2.6: Question types in IconQA.

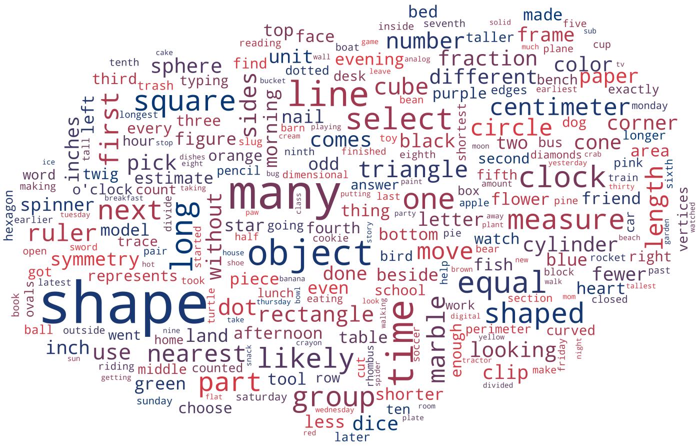


Figure 2.7: Word cloud of the question text in IconQA.

tion distribution implies the requirement of high-level understanding of textual and visual contents in IconQA. Figure 2.7 shows the word cloud of the question text in IconQA after eliminating the stop words. The most frequent words: *shape*, *many*, and *object* indicate that answering IconQA questions requires the model to identify a variety of geometric shapes and icon objects. Inspired by this, learning informative representations for icon images plays an important role in visual reasoning for the IconQA task.

Skill types	**Description**
Geometry	Identify shapes, symmetry, transformations
Counting	Count objects, shapes
Comparing	Compare object attributes
Spatial	Identify spatial positions and relations
Scene	Understand abstract scenes
Pattern	Identify next and different patterns
Time	Identify time of clocks, events
Fraction	Perform fraction operations
Estimation	Estimate lengths, large numbers
Algebra	Perform algebraic operations
Measurement	Measure widths, lengths, heights
Commonsense	Apply external knowledge
Probability	Perform probability and statistics operations

Table 2.6: Definition of reasoning skill types the in IconQA dataset.

Skill categories. Our IconQA dataset contains questions of multiple different cognitive reasoning and arithmetic reasoning types that can be grouped into 13 categories, shown in Table 2.6.

We annotate each question in IconQA with its corresponding skill types based on the tags provided by the original problem sources. Figure 2.8 shows the distributions of questions related to each skill. For instance, to answer 13.8% of the questions in IconQA, the model has to be capable of *comparing* object attributes.

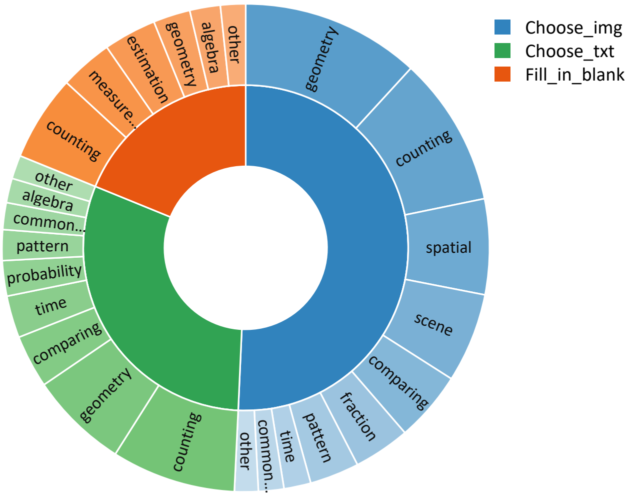


Figure 2.8: Skill distribution in IconQA questions.

Each question can be related to up to three skills out of these 13 categories, and on average, a question requires 1.63 skills. The detailed statistics are demonstrated in Table 2.7. In general, the *filling-in-the-blank* sub-task consists of questions that require the most number of skills, averaging 1.81 skills per question. 9.25% of the *filling-in-the-blank* questions require 3 skills.

Task	**Avg.**	**1 skill**	**2 skills**	**3 skills**
Multi-image-choice	1.51	55.78%	37.44%	6.77%
Multi-text-choice	1.73	33.21%	60.14%	6.65%
Filling-in-the-blank	1.81	28.30%	62.43%	9.25%
All	1.63	44.50%	48.34%	7.16%

Table 2.7: Skill numbers for questions in the IconQA dataset.

As the examples from IconQA shown in Figure 2.3, the first and second questions require the skills of *scene* understanding and *spatial* reasoning. The third example asks how many sticks exist in the diagram, requiring the basic ability of *counting* and basic *algebra* operations. As stated before, the IconQA dataset requires a wide range of skills for a model to perform well on IconQA.

	#QA	#Image	AvgQ	MaxQ	Image Type	Source	#Object	#Task	VisualAns	CommonSen	Arithmetic
VQA [AAL15]	614,163	204,721	6.1	23	Natural	Annotated	-	2		✓	
CLEVR [JHM17]	999,968	100,000	18.4	43	Natural	Generated	3	1			
VQA-Abstract [AAL15]	150,000	50,000	6.0	21	Scene	Annotated	131	2			
DVQA [KPC18]	2,325,316	300,000	10.3	23	Bar chart	Generated	-	1			✓
NLVR [SLY17]	92,244	92,244	11.2	25	Scatter plot	Generated	3	1			
Geometry3K [LGJ21]	3,002	2,342	10.1	46	Diagram	Real-world	4	1			✓
AI2D [KSK16]	4,563	4,903	9.8	64	Illustration	Real-world	-	1		✓	
IconQA (Ours)	107,439	96,817	8.4	73	Icon image	Real-world	388	3	✓	✓	✓

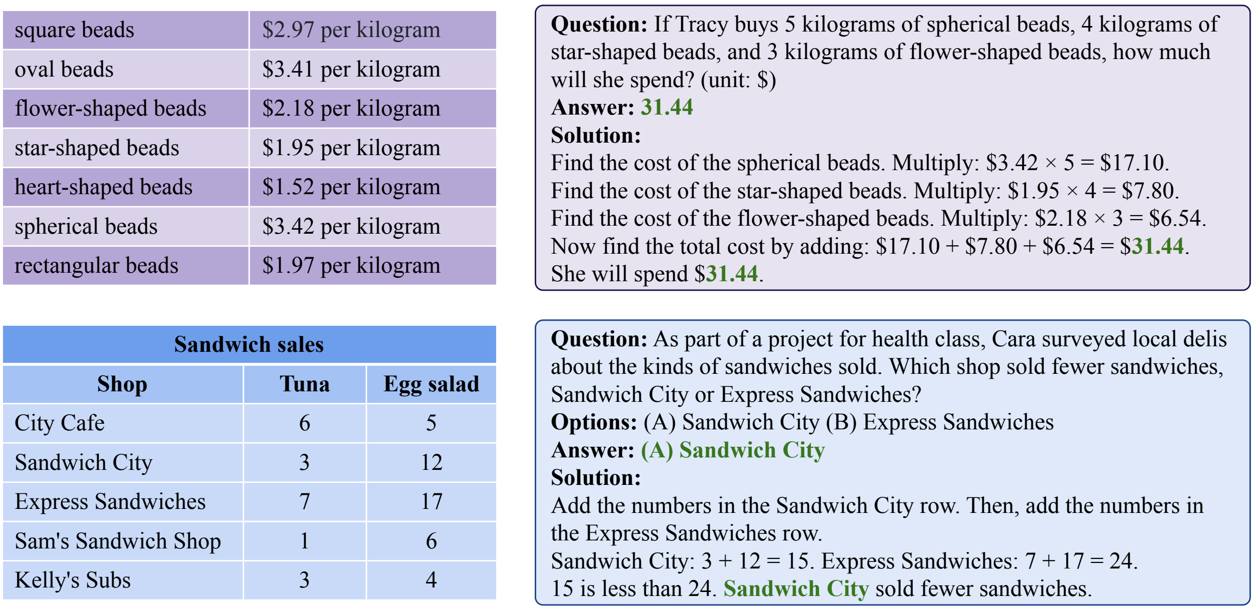
Table 2.8: Statistics for the IconQA dataset and comparisons with existing datasets.

Comparisons to other datasets. We compare our IconQA dataset with two datasets on natural images and five datasets on abstract diagrams in Table 2.8. To summarize, IconQA is different from these datasets in various aspects. Unlike natural images (VQA [AAL15], CLEVR [JHM17]) or abstract diagrams like scenes, charts, plots, and illustrations (VQA-Abstract [AAL15], DVQA [KPC18], NLVR [SLY17], AI2D [KSK16], Geometry3K [LGJ21]), IconQA features icon images and covers the largest object set of 388 classes. As questions in IconQA stem from real-world math problems and they may describe complex problem scenarios, IconQA has the longest question length among all related datasets. Furthermore, IconQA requires both commonsense and arithmetic reasoning due to its origin from real-world problems. Lastly, IconQA contains more QA task types including answering questions with image choices.

2.3 Math Word Problems in Tabular Contexts

2.3.1 Introduction

Developing machines equipped with mathematical reasoning capabilities is one of the long-standing goals of artificial intelligence. Solving math word problems (MWPs) is a well-defined task to diagnose the ability of intelligent systems to perform numerical reasoning and problem-solving as humans. A surge of datasets has been proposed to facilitate the research in this domain [UC17, AGL19, MLS20, CKB21]. However, most existing MWP datasets focus on textual math word problems only. Tables, widely distributed in different documents such as invoices, health records, and financial reports, contain rich structured information different from unstructured text. Solving math word problems in such a tabular context is much more challenging than existing MWP



square beads	$2.97 per kilogram
oval beads	$3.41 per kilogram
flower-shaped beads	$2.18 per kilogram
star-shaped beads	$1.95 per kilogram
heart-shaped beads	$1.52 per kilogram
spherical beads	$3.42 per kilogram
rectangular beads	$1.97 per kilogram

Question: If Tracy buys 5 kilograms of spherical beads, 4 kilograms of star-shaped beads, and 3 kilograms of flower-shaped beads, how much will she spend? (unit: $)
Answer: **31.44**
Solution:
Find the cost of the spherical beads. Multiply: $3.42 × 5 = $17.10.
Find the cost of the star-shaped beads. Multiply: $1.95 × 4 = $7.80.
Find the cost of the flower-shaped beads. Multiply: $2.18 × 3 = $6.54.
Now find the total cost by adding: $17.10 + $7.80 + $6.54 = $**31.44**.
She will spend $**31.44**.

Sandwich sales

Shop	**Tuna**	**Egg salad**
City Cafe	6	5
Sandwich City	3	12
Express Sandwiches	7	17
Sam's Sandwich Shop	1	6
Kelly's Subs	3	4

Question: As part of a project for health class, Cara surveyed local delis about the kinds of sandwiches sold. Which shop sold fewer sandwiches, Sandwich City or Express Sandwiches?
Options: (A) Sandwich City (B) Express Sandwiches
Answer: **(A) Sandwich City**
Solution:
Add the numbers in the Sandwich City row. Then, add the numbers in the Express Sandwiches row.
Sandwich City: 3 + 12 = 15. Express Sandwiches: 7 + 17 = 24.
15 is less than 24. **Sandwich City** sold fewer sandwiches.

Figure 2.9: Two examples from the TabMWP dataset. The top is a *free-text* problem with a numerical answer; the bottom is a *multi-choice* problem with a textual answer.

benchmarks since the system needs to make cell selections and align heterogeneous information before performing further numerical reasoning.

To fill this gap, we propose Tabular Math Word Problems (TabMWP), a new large-scale dataset that contains 38,431 math word problems with tabular context, taken from grade-level math curricula. There are two question types: *free-text* questions in which the answer is an integer or decimal number, and *multi-choice* questions where the answer is a text span chosen from option candidates. Different from existing MWP datasets, each problem in TabMWP is accompanied by a tabular context, which is represented in three formats: an image, a semi-structured text, and a structured table. Each problem is also annotated with a detailed solution that reveals the multi-step reasoning steps to ensure full explainability.

To solve problems in TabMWP, a system requires multi-hop mathematical reasoning over heterogeneous information by looking up table cells given textual clues and conducting multi-step operations to predict the final answer. Take the problem above in Figure 2.9 as an example. To answer the question "*how much will she spend (if Tracy buys three kinds of beads)*?", we first need to look up the corresponding three rows in the given table, calculate the individual cost for each kind of bead, and finally sum three costs up to get the answer of 31.44.

2.3.2 Related Datasets

Math word problem datasets. The task of solving MWPs is to predict the answer given a natural language description of a math problem. There have been great efforts in developing datasets for MWPs, including Math23K [WLS17], MathQA [AGL19], ASDiv [MLS20], SVAMP [PBG21], and Lila [MFL22]. However, these datasets only involve the textual modality, and most are limited to a small data scale. Some recent datasets like DVQA [KPC18], IconQA [LQC21], Geometry3K [LGJ21], and UniGeo [CLQ22] introduce math problems with diagrams as the visual context, where the system needs to perform mathematical reasoning over multi-modal information. To the best of our knowledge, our dataset TabMWP is the first dataset that requires mathematical reasoning over heterogeneous information from both the textual question and the tabular context.

Table QA datasets. Table Question Answering (Table QA) refers to the task of answering questions about tabular data. Numerous datasets have been developed for Table QA. For example, TabMCQ [JTH16] is an early dataset collected from grade exams. Datasets like WTQ [PL15], WikiSQL [ZXS17], and SQA [IYC17] contain semi-structured tables from Wikipedia, while Spider [YZY18] collects structured tables sourced from databases. Recent work aims at introducing datasets that require multi-hop reasoning between the textual and tabular data [CZC20, CCS20, TYC20, KCK21, NHM22]. Datasets most related to our TabMWP dataset are FinQA [CCS21], TAT-QA [ZLH21], and MultiHiertt [ZLL22] because they need numerical reasoning on financial reports with tabular data. Note that 77.6% of questions in TAT-QA can be solvable without mathematical reasoning and 50.0% of questions in FinQA are not table-must to be answered. In contrast, our proposed TabMWP collects questions where both mathematical reasoning and tabular context are necessary, and each question is annotated with a solution in natural language.

2.3.3 Dataset Collection

Data source. We construct TabMWP based on openly available content. The raw problems are collected from an online learning website, IXL[5], which hosts a large number of high-quality math

---

[5] `https://www.ixl.com/math`

problems curated by educational experts. Only math word problems that are accompanied by a tabular context and a detailed solution are collected. We develop a script to extract the tabular context, the question, options that apply, the correct answer, and the solution for each problem. These elements can be precisely identified using HTML tags. For each table, we take a screenshot and store its raw text.

Data preprocessing. To make TabMWP compatible with various baselines, we represent the tabular context as three formats: an image, *semi-structured* text, and a *structured* spreadsheet. The semi-structured format is created by converting the raw table text into a flattened token sequence, with each row separated by a newline character '`\n`' and each column separated by '|'. The semi-structured text is further transformed to the structured format, which can be easily retrieved and executed by SQL-based methods [LCG22] using packages like `pandas`. For clarity, the table title is separated from the raw table. Examples of three formats are shown in Table 2.9.

Quality control. We further conduct quality control to ensure data quality. The goal of constructing TabMWP is to collect math word problems that necessitate multi-hop mathematical reasoning between the question and the tabular context. Therefore, we ask human experts to filter problems that can be solved either without the context of the table or by looking up table cells without numerical reasoning. To further ensure data quality, we ask human experts to perform a final review to re-check the dataset and manually revise incorrect annotations.

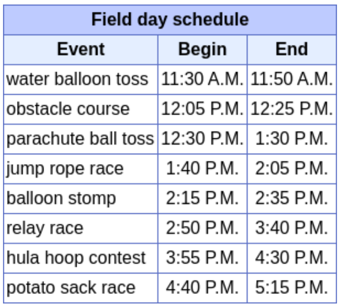
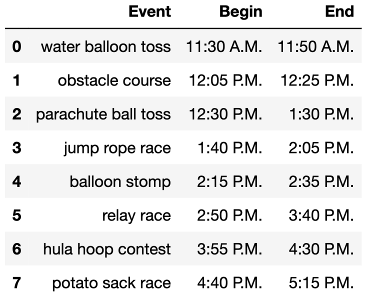
<table>
<tr><th>Image format</th><th>Semi-structured format</th><th>Structured format</th></tr>
<tr>
<td>

<table>
<tr><th colspan="3">Field day schedule</th></tr>
<tr><th>Event</th><th>Begin</th><th>End</th></tr>
<tr><td>water balloon toss</td><td>11:30 A.M.</td><td>11:50 A.M.</td></tr>
<tr><td>obstacle course</td><td>12:05 P.M.</td><td>12:25 P.M.</td></tr>
<tr><td>parachute ball toss</td><td>12:30 P.M.</td><td>1:30 P.M.</td></tr>
<tr><td>jump rope race</td><td>1:40 P.M.</td><td>2:05 P.M.</td></tr>
<tr><td>balloon stomp</td><td>2:15 P.M.</td><td>2:35 P.M.</td></tr>
<tr><td>relay race</td><td>2:50 P.M.</td><td>3:40 P.M.</td></tr>
<tr><td>hula hoop contest</td><td>3:55 P.M.</td><td>4:30 P.M.</td></tr>
<tr><td>potato sack race</td><td>4:40 P.M.</td><td>5:15 P.M.</td></tr>
</table>
</td>
<td>
<b>Table title</b>: Field day schedule
<b>Table text</b>:
Event | Begin | End
water balloon toss | 11:30 A.M. | 11:50 A.M.
obstacle course | 12:05 P.M. | 12:25 P.M.
parachute ball toss | 12:30 P.M. | 1:30 P.M.
jump rope race | 1:40 P.M. | 2:05 P.M.
balloon stomp | 2:15 P.M. | 2:35 P.M.
relay race | 2:50 P.M. | 3:40 P.M.
hula hoop contest | 3:55 P.M. | 4:30 P.M.
</td>
<td>
<b>Table title</b>: Field day schedule

<table>
<tr><th></th><th>Event</th><th>Begin</th><th>End</th></tr>
<tr><td>0</td><td>water balloon toss</td><td>11:30 A.M.</td><td>11:50 A.M.</td></tr>
<tr><td>1</td><td>obstacle course</td><td>12:05 P.M.</td><td>12:25 P.M.</td></tr>
<tr><td>2</td><td>parachute ball toss</td><td>12:30 P.M.</td><td>1:30 P.M.</td></tr>
<tr><td>3</td><td>jump rope race</td><td>1:40 P.M.</td><td>2:05 P.M.</td></tr>
<tr><td>4</td><td>balloon stomp</td><td>2:15 P.M.</td><td>2:35 P.M.</td></tr>
<tr><td>5</td><td>relay race</td><td>2:50 P.M.</td><td>3:40 P.M.</td></tr>
<tr><td>6</td><td>hula hoop contest</td><td>3:55 P.M.</td><td>4:30 P.M.</td></tr>
<tr><td>7</td><td>potato sack race</td><td>4:40 P.M.</td><td>5:15 P.M.</td></tr>
</table>
</td>
</tr>
</table>

Table 2.9: Three different formats for the tables in the TabMWP dataset.

For better quantitative evaluation, we formalize the TabMWP problems as two question types: (a) *free-text* questions, where the answer is numerical text only and the unit text is separately

Question types	Answer types (%)	Descriptions
Free-text	Integer (59.50%)	The answer is an integer number, e.g., “40”, “1,207”, “-3”.
	Decimal (15.23%)	The answer is a decimal or a fraction number, e.g., “192.80”, “68/217”.
Multi-choice	Extractive (13.01%)	The answer could be extracted from the table context.
	Boolean (10.97%)	The answer is Boolean, e.g., “yes”/“no”, “true”/“false”, “linear”/“nonlear”.
	Other (1.29%)	The answer belongs to other text types, e.g., a statement.

Table 2.10: Format diversity of questions and answers in the TabMWP dataset.

Statistic	Number
Total questions	38,431
* *free-text* questions	28,719
* *multi-choice* questions	9,712
# of different questions	28,876
# of different answers	6,153
# of different solutions	35,442
# of different tables	37,644
# of tables with a title	23,259
# of table cells (Average/Max)	12.9 / 54
# of table rows (Average/Max)	5.9 / 11
# of table columns (Average/Max)	2.2 / 6
Question length (Average/Max)	22.1 / 92
Answer length (Average/Max)	1.1 / 27
Solution length (Average/Max)	49.5 / 350

Table 2.11: Key statistics for the TabMWP dataset.

extracted; and (b) *multi-choice* questions, the answer of which is the text span from choice options, as defined in Table 2.10. The order of choice options is shuffled to alleviate distribution bias. Redundant information in solutions is removed, and some solutions are manually rewritten to be more human-readable. Finally, problems with the same table, question, and answer text are regarded as redundant and thus removed.

2.3.4 Dataset Statistics

Key statistics. The TabMWP dataset contains 38,431 tabular math word problems, which are partitioned with 6:2:2 into the training, development, and test splits, corresponding to 23,059, 7,686, and 7,686 problems. Their main statistics are shown in Table 2.11. 74.7% of the questions in TabMWP belong to *free-text* questions, while 25.3% are *multi-choice* questions. There are 28,876 different questions, 6,153 different answers, and 35,442 different solutions, indicating that TabMWP has a rich diversity in the problem distribution. The questions have an average of 22.1 words in length and solutions of 49.5, showing that they have lexical richness.

One distinct characteristic of TabMWP is that each problem is accompanied by a tabular context, without which the problem would be unsolvable. There are 37,644 different tables in total, and 60.5% of the tables have a title. The table has an average of 5.9 rows and 2.2 columns, which results in an average of 12.9 cells and a maximum of 54 cells. These statistics suggest that tables in TabMWP distribute diversely across semantics and layouts.

Dataset	**Size**	**#Table**	**Need Math?**	**Need Table?**	**Table Type**		**Question Type**		**Answer Type**			**Solution Type**
					Domain	**Format**	**Free-text**	**MC**	**Text**	**Integer**	**Decimal**	
Dolphin18K [HSL16]	831	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓	formula
DRAW-1K [UC17]	1,000	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓	formula
Math23K [WLS17]	23,162	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓	formula
MathQA [AGL19]	37,297	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓	formula
ASDiv [MLS20]	2,305	✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	formula
SVAMP [PBG21]	1,000	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	formula
GSM8K [CKB21]	8,792	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	text
IconQA [LQC21]	107,439	✗	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗
FinQA [CCS21]	8,281	2,766	✓	76.6%	finance	text	✓	✗	✗	✓	✓	program
TAT-QA [ZLH21]	16,552	2,747	50.0%	✓	finance	text	✓	✗	✗	✓	✓	✗
MultiHiertt [ZLL22]	10,440	9,843	✓	89.8%	finance	text	✓	✗	✗	✓	✓	✗
TabMWP (ours)	**38,431**	**37,644**	✓	✓	**open**	**text***	✓	✓	✓	✓	✓	**text**

Table 2.12: A comparison of MWP and Table QA datasets that require numerical reasoning. *text**: each table in TabMWP is accompanied by an image format.

Comparison to existing datasets. As shown in Table 2.12, TabMWP differs from related datasets in various aspects: (1) TabMWP is the first dataset to study math word problems over tabular context on open domains and is the largest in terms of data size; (2) Problems in TabMWP are annotated with the tabular context, unlike previous MWP datasets in the first segment; (3) Different from Table QA datasets like FinQA, TAT-QA, and MultiHiertt, a lack of either mathematical reasoning or the tabular context renders the problems in TabMWP unanswerable; (4) There are two question types in TabMWP, and the answer could be a text span, an integer number, or a decimal number; (5) Each problem is annotated with natural language solutions to reveal multi-hop reasoning steps.

CHAPTER 3

Knowledge-Intensive Scientific Reasoning

3.1 Introduction

A long-standing goal of AI systems is to act reliably and learn complex tasks efficiently like human beings. In the process of reliable decision making, humans follow an explicit *chain-of-thought* (CoT) reasoning process that is typically expressed as an explanation. However, machine learning models are trained mostly using a large number of input-output examples to perform a specific task. These black-box models only generate the final decision without reliably revealing the underlying reasoning process. Not surprisingly, it is unclear if they understand the task and can generalize even though they perform well on the benchmark. On the other hand, humans are able to learn from instructions or explanations from past experience and generalize them to novel and unseen problems. This helps them learn more quickly with fewer data. In this section, we explore if machines can be endowed with such reasoning abilities in the context of science-based question answering.

Recently, science problem solving benchmarks [KSS17] have been used to diagnose the multi-hop reasoning ability and interpretability of AI systems. To answer science questions, a model needs to not only understand multimodal contents but also extract external knowledge to arrive at the correct answer. Since these tasks require domain-specific knowledge and explicit multi-hop reasoning, a model would be not interpretable if it fails to provide explanations to reveal the reasoning process. However, current science question datasets [KSS17, KSK16, SYB20] mostly lack annotated explanations for the answers. To address this issue, other science datasets annotate the explanations, but they are restricted to the textual only modality and limited to small data scales [JWM18, DJT21, MCK18] or a small set of topics [KCG20, JC20]. Therefore, we collect Science

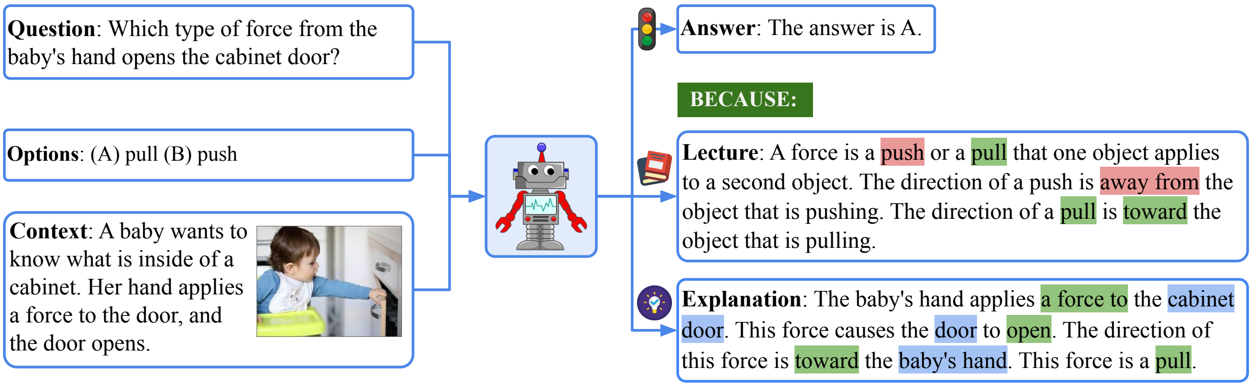

Question: Which type of force from the baby's hand opens the cabinet door?
Options: (A) pull (B) push
Context: A baby wants to know what is inside of a cabinet. Her hand applies a force to the door, and the door opens.
Answer: The answer is A.
BECAUSE:
Lecture: A force is a push or a pull that one object applies to a second object. The direction of a push is away from the object that is pushing. The direction of a pull is toward the object that is pulling.
Explanation: The baby's hand applies a force to the cabinet door. This force causes the door to open. The direction of this force is toward the baby's hand. This force is a pull.

Figure 3.1: We construct the ScienceQA dataset where a data example consists of multimodal question answering information and the grounded lecture and explanation.

Question Answering (ScienceQA), a large-scale multi-choice dataset that contains multimodal science questions with explanations and features rich domain diversity.

ScienceQA is collected from elementary and high school science curricula, and contains 21,208 examples along with lectures and explanations. Different from existing datasets [KSK16, KSS17, SYB20], ScienceQA has richer domain diversity from three different subjects: natural science, social science, and language science. A typical example consists of a question, multiple choices, multimodal contexts, a correct answer, as well as a lecture and an explanation. The lecture and explanation provide general external knowledge and specific reasons, respectively, for arriving at the correct answer.

Consider the thoughts one person might have when answering the question in Figure 3.1. One first recalls the knowledge regarding the definition of a force learned from textbooks: "*A force is a push or a pull that ... The direction of a* ***push*** *is ... The direction of a* ***pull*** *is ...*", then forms a line of reasoning: "*The baby's* ***hand*** *applies a force to the cabinet* ***door***. → *This force causes the* ***door*** *to* ***open***. → *The direction of this force is* ***toward*** *the baby's* ***hand***.", and finally arrives at the correct answer: "*This force is a* ***pull***.". Following [NRL20], we formulate the task to output a natural explanation alongside the predicted answer. In this work, we train language models to generate lectures and explanations as the *chain of thought* (CoT) to mimic the multi-hop reasoning process to answer ScienceQA questions.

Our experiments show that current multimodal methods [YYC19, AHB18, KJZ18, GJY19, LYY19, LQC21] fail to achieve satisfactory performance on ScienceQA and do not generate correct explanations. Instead, we find that CoT can help large language models not only in the few-shot learning setting but also in the fine-tuning setting. When combined with CoT to generate the lecture and explanation, the fine-tuned UnifiedQA [KMK20] achieves an improvement of 3.99% as opposed to not using CoT in the fine-tuning stage. The few-shot GPT-3 model [BMR20] via chain-of-thought prompting can obtain 75.17% on ScienceQA with an improvement of 1.20% compared to the few-shot GPT-3 without CoT. Prompted with CoT, GPT-3 can generate reasonable explanations as evaluated by automated metrics, and promisingly, 65.2% of explanations meet the gold standard of human evaluations. We also investigate the upper bound for models to harness explanations by including them in the input. We find that doing so improves GPT-3's few-shot performance by 18.96%, suggesting that explanations do aid models and are currently underutilized in the CoT framework. Further analysis shows that, like humans, language models benefit from explanations to learn with less data: UnifiedQA with CoT obtains the same results as UnifiedQA without CoT with only 40% of the training data.

To sum up, our contributions are three-fold: (a) To bridge the gap in existing datasets in the scientific domain, we build Science Question Answering (ScienceQA), a new dataset containing 21,208 multimodal science questions with rich domain diversity. To the best of our knowledge, ScienceQA is the first large-scale multimodal dataset that annotates lectures and explanations for the answers. (b) We show that CoT benefits large language models in both few-shot and fine-tuning learning by improving model performance and reliability via generating explanations. (c) We further explore the upper bound of GPT-3 and show that CoT helps language models learn from fewer data.

3.2 Related Work

Visual question answering. Since the task of visual question answering (VQA) was first proposed in [AAL15], there have been plenty of VQA datasets [ZGS16, ZGB16, KZG17, GKS17, JHM17, HM19] conducted to facilitate the research work. Although our ScienceQA dataset shares some features with VQA, there are several main differences between them. First, ScienceQA is more challenging than existing VQA datasets because it contains multimodal contexts and diverse topics in the scientific domain. In addition, most answers are annotated with lectures and explanations, which makes ScienceQA a suitable dataset for multi-modal question answering and multi-hop reasoning for AI systems. Inspired by the recent remarkable performance achieved for VQA [LLZ18, LJZ18, GLL18, GJY19, LYY19, DBK21], in this work, we further extensively benchmark ScienceQA with a wide range of attention-based [AHB18, LLZ18, KJZ18, GJY19] and Transformer-based [LBP19, LYY19, LYY20, DBK21] methods.

Datasets for science problems. Science problem solving is a challenging task that requires an AI system not only to understand the multimodal information from the science curriculum but also to reason about how to answer the domain-specific questions. Current science problem datasets such as AI2D [KSK16], DVQA [KPC18], VLQA [SYB20], and FOODWEDS [KTK16] have contributed to multimodal reasoning in the scientific domain. For example, a portion of VLQA contains multimodal questions on science subjects. These datasets, however, lack annotated explanations for the answers to reveal the reasoning steps. Some other datasets annotate the answers in the forms of supporting facts [MCK18, KCG20], entailment trees [DJT21], explanation graphs [JWM18], reasoning chains [JC20]. However, these datasets are restricted to the single text modality with small data scales and limited topics. Instead, our ScienceQA annotates the answers with grounded lectures and explanations. Besides, ScienceQA features a richer domain diversity across 3 subjects, 26 topics, 127 categories, and 379 skills.

Learning from explanations and few-shot learning. Explanations help humans understand a task better, and there have been several attempts to show the same for models. For example, the learning from instruction paradigm [MKB21, OWJ22, WBZ21, MKB22, PMP22, LDC22], where the task level explanation is provided in the form of instruction, improves model performance significantly. An example of learning from explanations in the scientific domain is proposed in [SX17] where the model interprets demonstrative solutions to solve geometry problems. Recently, there has been a surge of interest in few-shot learning, where language models learn a specific task from a few examples [PKC21, BCL21]. For instance, [NAG22, WWS22b, LQC23] find that explanations in the format of the chain of thought can improve language models' reasoning ability in few-shot learning. In this work, we show that the chain of thought boosts the performance of large language models like UnifiedQA [KMK20] if the models generate explanations along with the answer in a fine-tuning way. Furthermore, a few-shot GPT-3 model via chain-of-thought prompting is able to improve the reasoning performance on ScienceQA and generate reasonable explanations.

3.3 Dataset

We collect ScienceQA, which is a multimodal multiple-choice science question dataset containing 21,208 examples. An example in ScienceQA is shown in Figure 3.1. Given the science question and multimodal contexts, the task is to select the correct answer from multiple options. Different from existing datasets [SDX17, KSK16, SYB20, LGJ21, KTK16], ScienceQA covers diverse topics across three subjects: natural science, social science, and language science. Moreover, most questions are annotated with grounded lectures and detailed explanations. The lecture provides general knowledge that introduces the background information for solving problems of a similar class. The explanation reveals a specific reason for the answer. To effectively answer the questions, a model often needs to be able to understand the multimodal content in the input and extract external knowledge, similar to how humans do. More importantly, the goal of ScienceQA is to aid development of a reliable model that is capable of generating a coherent chain of thought when arriving at the correct answer to reveal the multi-step reasoning process.

3.3.1 Data Collection

Questions in the ScienceQA dataset are sourced from open resources managed by IXL Learning, an online learning platform curated by experts in the field of K-12 education. The dataset includes problems that align with *California Common Core Content Standards*. To construct ScienceQA, we downloaded the original science problems and then extracted individual components (e.g. questions, hints, images, options, answers, lectures, and solutions) from them based on heuristic rules.

We manually removed invalid questions, such as questions that have only one choice, questions that contain faulty data, and questions that are duplicated, to comply with *fair use* and *transforma-*

Statistic	**Number**
Total questions	21,208
Questions with text context	10,220 (48.2%)
Questions with image context	10,332 (48.7%)
* Image of natural format	≈2,960 (14.0%)
* Image of diagram format	≈7,372 (34.8%)
Questions with both contexts	6,532 (30.8%)
Questions with a lecture	17,798 (83.9%)
Questions with a explanation	19,202 (90.5%)
Different questions	9,122
Different lectures	261
Topic classes	26
Category classes	127
Skill classes	379
Average question length	12.11
Average choice length	4.40
Average lecture length	125.06
Average explanation length	47.66

Table 3.1: Main statistics of the ScienceQA dataset.

tive use of the law. If there were multiple correct answers that applied, we kept only one correct answer. Also, we shuffled the answer options of each question to ensure the choices do not follow any specific pattern. To make the dataset easy to use, we then used semi-automated scripts to reformat the lectures and solutions. Therefore, special structures in the texts, such as tables and lists, are easily distinguishable from simple text passages. Similar to ImageNet, ReClor, and PMR datasets, ScienceQA is available for non-commercial research purposes only and the copyright belongs to the original authors. To ensure data quality, we developed a data exploration tool to review examples in the collected dataset, and incorrect annotations were further manually revised by experts. The tool can be accessed at `https://scienceqa.github.io/explore.html`.

3.3.2 Data Analysis

Key statistics. We randomly split the dataset into training, validation, and test splits with a ratio of 60:20:20. Each split has 12,726, 4,241, and 4,241 examples, respectively. Table 3.1 shows the main statistics of ScienceQA. ScienceQA has a large set of different questions, totaling up to

9,122. Out of the 21,208 questions in ScienceQA, 10,332 (48.7%) have an image context, 10,220 (48.2%) have a text context, and 6,532 (30.8%) have both. 83.9% of the questions are annotated with a lecture, while 90.5% of the questions feature an explanation. The cross-combination of these information sources diversifies the problem scenario: sometimes the model is given a lot of information from multiple sources, while at other times, the only source of information is the question itself. This level of complexity is very common in grade-level science exams.

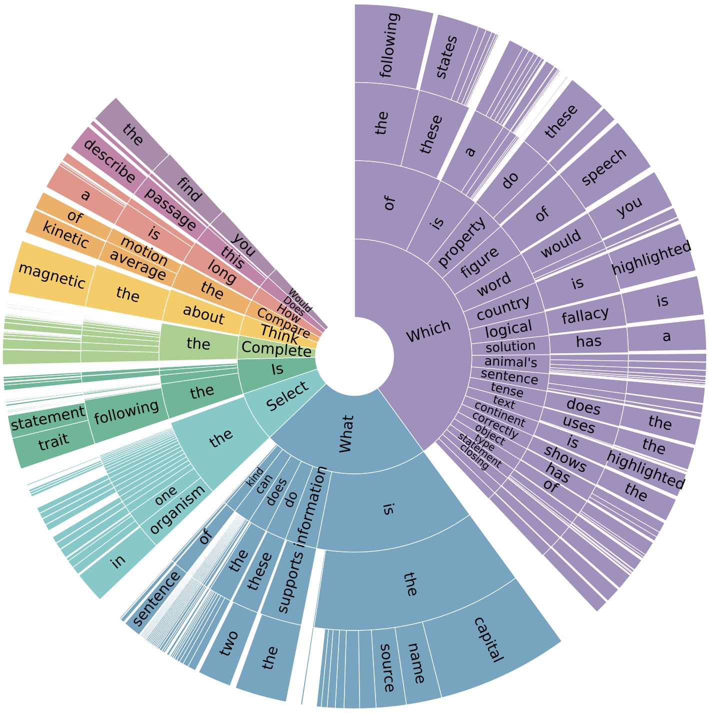


Figure 3.2: Question distribution of the ScienceQA dataset.

Question analysis. ScienceQA has a diverse set of science questions. Figure 3.2 shows a distribution of the first four words in the question text. A large number of question lengths and formats highlight the diversity of ScienceQA. The question lengths range from 3 words to 141 words, and the questions in ScienceQA have an average length of 12.11 words. The question length distribution is visualized against other VQA datasets in Figure 3.3. As shown in the diagram, ScienceQA's distribution is flatter than other datasets, spanning more evenly across different question lengths.

Figure 3.4 (a) is a word cloud showing the most frequently appeared words in the question texts. Stopping words that do not contain any semantic meaning, such as "*what*" or "*and*", are

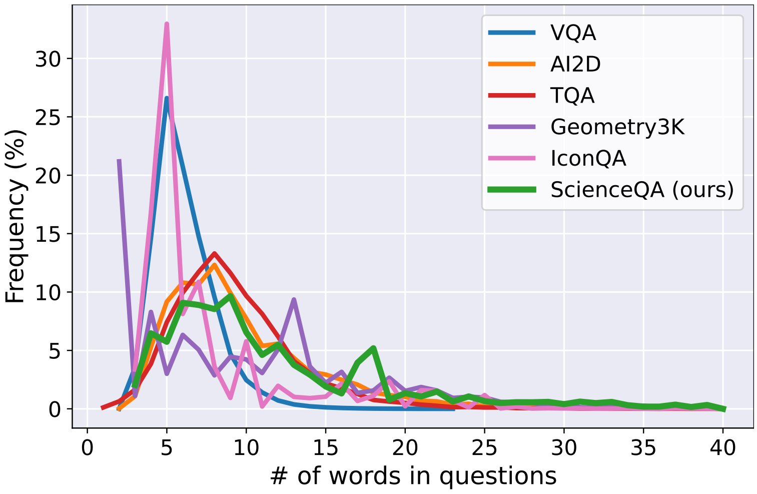


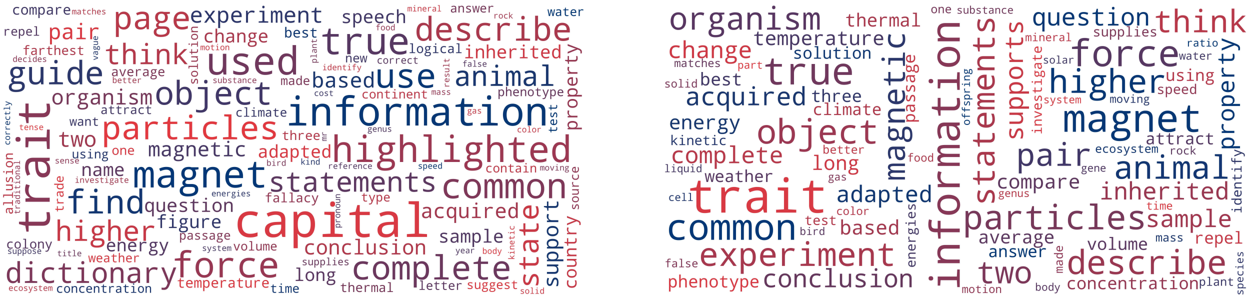
Figure 3.3: Question length distribution of related datasets in ScienceQA.

removed to give us a clearer view of the semantic range of ScienceQA. The diagram shows that ScienceQA covers a wide range of topics, with words from different topics showing up across the cloud.

Figures 3.4 (b) (c) (d) show the word clouds for each of the three subjects. We can observe from the word clouds that the words are well-matched to the subject themes. In natural science questions, words such as "*trait*", "*magnet*", and "*force*" appear frequently. Words such as "*capital*" and "state" show up frequently in social science questions, whereas words such as "*dictionary*" and "*page*" are common in language science questions.

Context analysis. Figure 3.5 shows the number and percentage of questions with either an image context, a text context, or both. There are a total of 7,803 unique image contexts and 4,651 unique text contexts. 66.11% of the questions have at least one type of context information. The image context is in the format of diagrams or natural images, which visualize the critical scenario necessary for question answering or simply illustrate the question for better understanding. Similarly, the textual context can provide either semantically rich information or a simple hint to the question. Therefore, models need to be flexible and general to understand these diverse types of contexts.

Domain diversity. Each ScienceQA question belongs to one of the three subjects: natural science, social science, and language science. With each subject, questions are categorized first by the topic (*Biology*, *Physics*, *Chemistry*, etc.), then by the category (*Plants*, *Cells*, *Animals*, etc.), and finally



(a) Questions of all subjects.

(b) Questions in natural science.

colony
north
ocean
result
farthest
highlighted
trade
country
state
continent
decides
cost
want
name
capital
suppose
letter
suggest
page
guide
fallacy
contain
allusion
use
find
correct
vague
used
figure
reference
logical
dictionary
speech
source
pronoun
title
sense

(c) Questions in social science.

(d) Questions in language science.

Figure 3.4: Word cloud distributions of question texts in different subjects in ScienceQA.

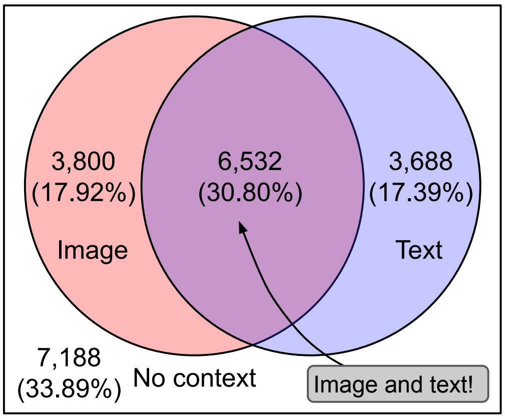
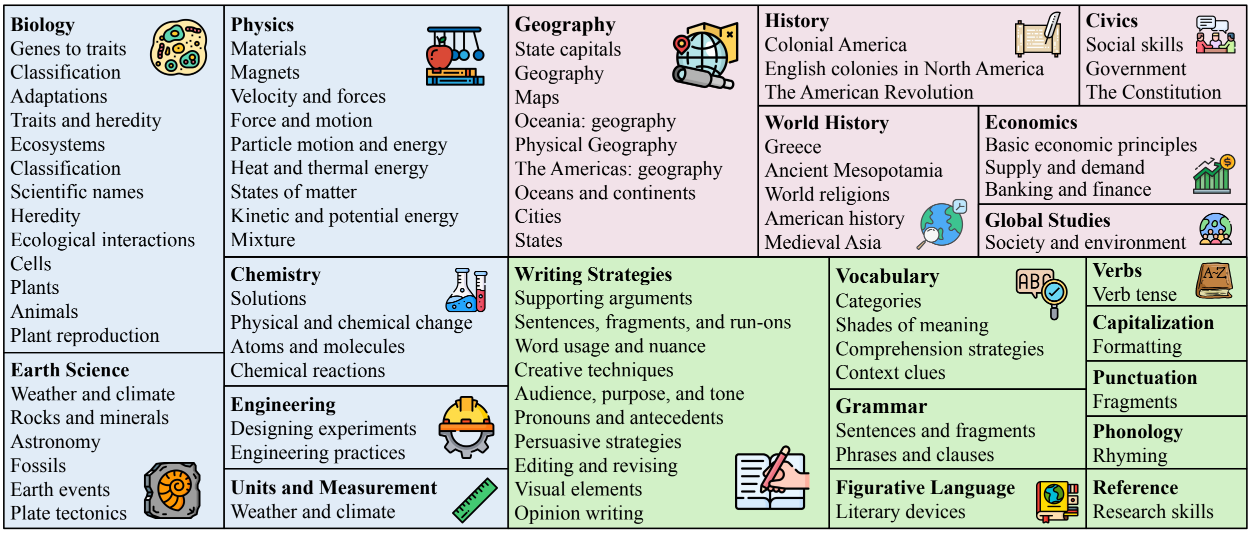


Figure 3.5: Question distribution with different context formats in ScienceQA.

by the specific skill (*Classify fruits and vegetables as plant parts*, *Identify countries of Africa*, etc.). ScienceQA has a total of 26 topics, 127 categories, and 379 skills. The treemap in Figure 3.6 visualizes the different subjects, topics, and categories and shows that ScienceQA questions are very diverse, spanning a wide range of domains.



Biology
Genes to traits
Classification
Adaptations
Traits and heredity
Ecosystems
Classification
Scientific names
Heredity
Ecological interactions
Cells
Plants
Animals
Plant reproduction

Earth Science
Weather and climate
Rocks and minerals
Astronomy
Fossils
Earth events
Plate tectonics

Physics
Materials
Magnets
Velocity and forces
Force and motion
Particle motion and energy
Heat and thermal energy
States of matter
Kinetic and potential energy
Mixture

Chemistry
Solutions
Physical and chemical change
Atoms and molecules
Chemical reactions

Engineering
Designing experiments
Engineering practices

Units and Measurement
Weather and climate

Geography
State capitals
Geography
Maps
Oceania: geography
Physical Geography
The Americas: geography
Oceans and continents
Cities
States

History
Colonial America
English colonies in North America
The American Revolution

Civics
Social skills
Government
The Constitution

World History
Greece
Ancient Mesopotamia
World religions
American history
Medieval Asia

Economics
Basic economic principles
Supply and demand
Banking and finance

Global Studies
Society and environment

Writing Strategies
Supporting arguments
Sentences, fragments, and run-ons
Word usage and nuance
Creative techniques
Audience, purpose, and tone
Pronouns and antecedents
Persuasive strategies
Editing and revising
Visual elements
Opinion writing

Vocabulary
Categories
Shades of meaning
Comprehension strategies
Context clues

Grammar
Sentences and fragments
Phrases and clauses

Figurative Language
Literary devices

Verbs
Verb tense

Capitalization
Formatting

Punctuation
Fragments

Phonology
Rhyming

Reference
Research skills

Figure 3.6: Domain diversity in ScienceQA. Each color corresponds to one subject: natural science, social science, and language science.

3.3.3 Choice Statistics

Table 3.2 shows the number of questions with each number of different choices. Questions have a minimum of two options and a maximum of five options. Figure 3.7 shows the distribution of choice length in ScienceQA. Most choices are short, containing up to five words. However, the distribution has a long tail where about 5% of the choices contain more than 15 words. Hence, it requires models to have a high level of text understanding to address diversely distributed choices.

Choice number	Size	Percent
2	11,045	52.08%
3	5,078	23.94%
4	4,893	23.07%
5	192	0.91%

Table 3.2: Choice number distribution in ScienceQA.

3.3.4 Grade Statistics

The grade distribution is shown in Table 3.8. The majority of questions come from the middle level curriculum (i.e., from grade 3 to grade 8) while around 10% are taken from the high school curriculum (i.e., from grade 9 to grade 12). These high school level questions are close to or at the

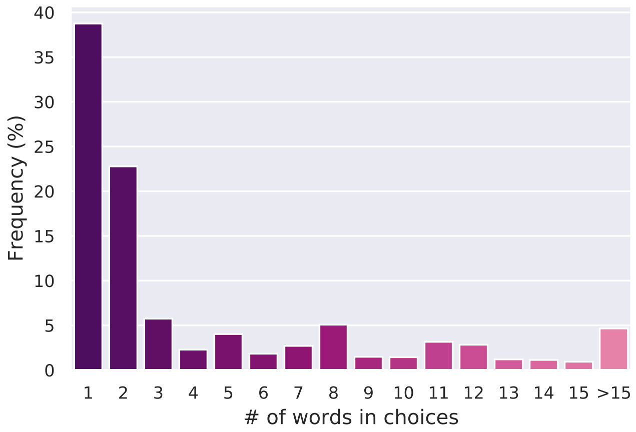


Figure 3.7: Choice length distribution in ScienceQA.

difficulty level of the U.S. standardized tests for college admissions. Machine algorithms need to master a large amount of scientific knowledge and perform complex reasoning in order to perform well on ScienceQA.

Grades	Number	Percent
Grade 1	95	0.45%
Grade 2	1,678	7.91%
Grade 3	3,032	14.3%
Grade 4	3,544	16.71%
Grade 5	3,086	14.55%
Grade 6	2,450	11.55%
Grade 7	2,749	12.96%
Grade 8	2,546	12.0%
Grade 9	491	2.32%
Grade 10	558	2.63%
Grade 11	539	2.54%
Grade 12	440	2.07%

Figure 3.8: Grade distribution statistics in ScienceQA.

3.3.5 Comparisons with Existing Datasets

Table 3.3 shows a comparison of ScienceQA and other science problem datasets. As shown in the table, ScienceQA is much larger than most other datasets. ScienceQA also has the largest set of images, spans across all 12 grades, contains the longest questions, and has the most diverse input sources. As opposed to limiting the subject to only natural science, ScienceQA also includes social science and language science, largely adding to the domain diversity of the dataset. Furthermore, most of the questions in ScienceQA are annotated with textual lectures (83.9%) and explanations (90.5%), which reveal the reasoning path to the correct answer. To the best of our knowledge, ScienceQA is the first large-scale multimodal science question dataset that annotates the answers with detailed lectures and explanations.

	#Q	**#I**	**AvgQ**	**MaxQ**	**Grades**	**Science subjects**	**Contexts**	**Images**	**Lecture**	**Explanation**
Geometry3K [LGJ21]	3,002	2,342	10.1	46	6-12	natural (geometry)	image	diagram	✗	✗
AI2D [KSK16]	4,563	4,903	9.8	64	1-6	natural	image	diagram	✗	✗
FOODWEBS [KTK16]	≈5,000	≈5,00	-	-	8	natural (foodweb only)	image	diagram	✗	✗
ARC [CCE18]	7,787	0	**20.4**	128	3-9	natural	✗	✗	✗	✗
TQA [KSS17]	**26,260**	3,455	9.2	57	6-8	natural	image, text	diagram	✓	✗
IconQA [LQC21]	107,439	96,817	8.4	73	PreK-3	math	visual	diagram	✗	✗
WorldTree [JWM18]	1,680	0	-	-	3-5	natural	✗	✗	✗	✓
OpenBookQA [MCK18]	5,957	0	10.6	68	1-6	natural	✗	✗	✗	✓
QASC [KCG20]	9,980	0	8.0	25	1-9	natural	✗	✗	✗	✓
ScienceQA (ours)	21,208	**10,332**	12.1	**141**	**1-12**	natural, social, language	image, text	natural, diagram	✓	✓

Table 3.3: Statistics for the ScienceQA dataset and comparisons with existing datasets.

3.4 Baselines and Chain-of-Thought Models

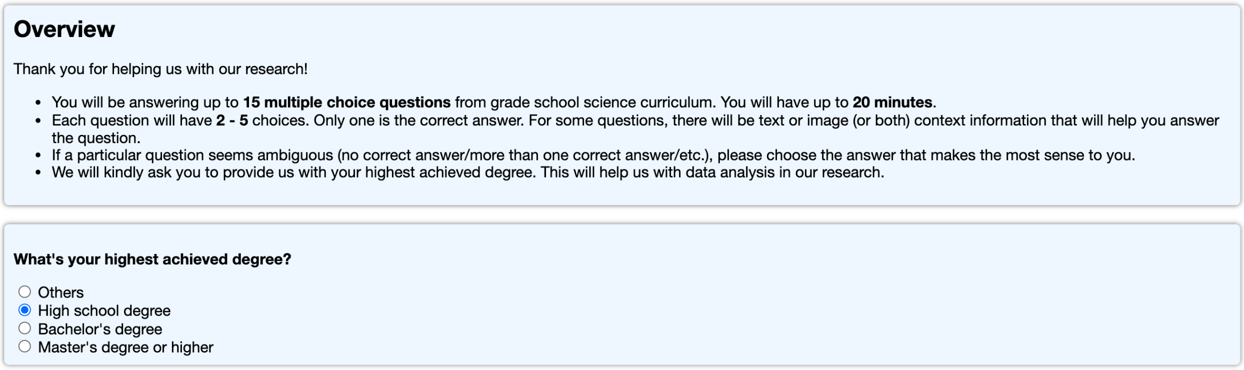
In this section, we establish baselines and develop two chain-of-thought models on ScienceQA.

3.4.1 Heuristic Baselines

The first heuristic baseline is *random chance*: we randomly select one from the multiple options. Each trial is completed on the whole test set, and we take three different trials for an average result. The second heuristic baseline is *human performance*. We post the task to Amazon Mechanical Turk and ask workers to answer ScienceQA questions. Only workers who obtain a high school or higher degree and pass the qualification examples are qualified for the study. Each worker needs to answer a set of 10 test questions, and each question is answered by three different workers.

In order to understand how humans perform on ScienceQA questions, we used Amazon Mechanical Turk (AMT) to crowdsource answers to the test set. The interface of instructions and one example of a test question is shown in Figures 3.9 and 3.10. A total of 4,241 test questions were shuffled and split into 425 batches, with each batch having 10 questions (excluding the last one).

For each batch, we also randomly added five training questions as exam examples. Each set of 15 questions was then assigned to 3 AMT workers. Only workers who correctly answer 4 out of the 5 exam examples or more are qualified for the human performance study. In other words, workers who failed to pass the qualified exam were eliminated from the analysis. For each set of 15 questions, we provided the worker with \$0.5 per HIT task. At the rate of 3 questions per minute, this amounts to \$6.0 per hour.



Overview

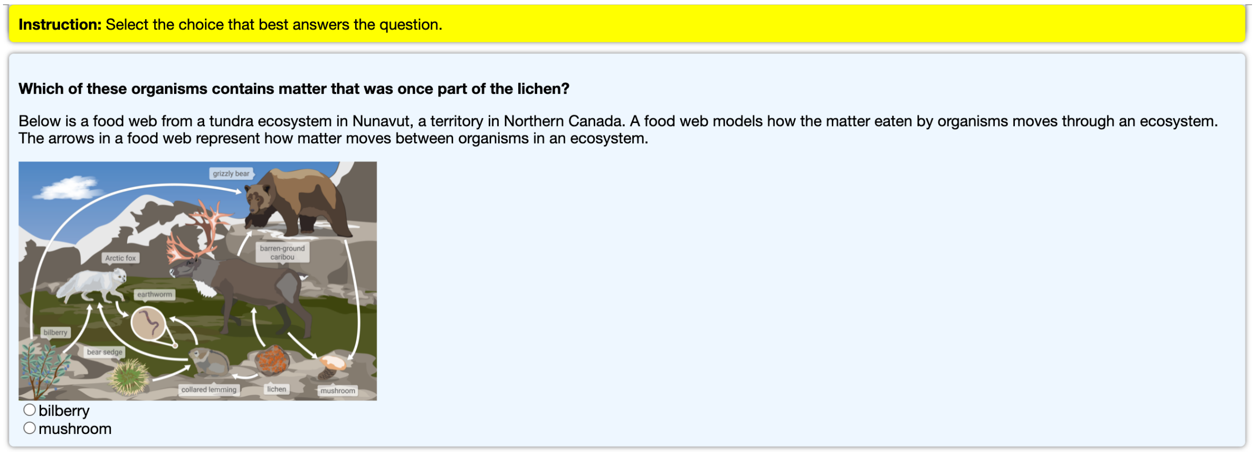
Thank you for helping us with our research!

- You will be answering up to **15 multiple choice questions** from grade school science curriculum. You will have up to **20 minutes**.
- Each question will have **2 - 5** choices. Only one is the correct answer. For some questions, there will be text or image (or both) context information that will help you answer the question.
- If a particular question seems ambiguous (no correct answer/more than one correct answer/etc.), please choose the answer that makes the most sense to you.
- We will kindly ask you to provide us with your highest achieved degree. This will help us with data analysis in our research.

What's your highest achieved degree?

○ Others
● High school degree
○ Bachelor's degree
○ Master's degree or higher

Figure 3.9: Instructions for AMT workers to answer the ScienceQA questions.



Instruction: Select the choice that best answers the question.

Which of these organisms contains matter that was once part of the lichen?

Below is a food web from a tundra ecosystem in Nunavut, a territory in Northern Canada. A food web models how the matter eaten by organisms moves through an ecosystem. The arrows in a food web represent how matter moves between organisms in an ecosystem.

○ bilberry
○ mushroom

Figure 3.10: A test question example from ScienceQA presented to AMT workers for evaluation.

3.4.2 Language Model Baselines

Zero-shot and few-shot baselines. We establish the zero-shot baselines on top of UnifiedQA [KMK20] and GPT-3 [BMR20]. The zero-shot setup follows the format of QCM→A where the input is the concatenation of tokens of the question text (Q), the context text (C), and multiple options (M), while the output is to predict the answer (A) from the option set. We extract the caption from the captioning model based on ViT [DBK21] and GPT-2 [RWC20] for the image as the visual context. In the few-shot setting, we follow the standard prompting [BMR20] where in-context examples from the training set are concatenated before the test instance. These in-context examples serve as an instruction for the language model to adjust to the specific task in ScienceQA.

Fine-tuning baselines. We first consider the fine-tuning baselines from VQA models [AHB18, KJZ18, YYC19, GJY19, KSK21, LQC21, LYY19] proposed in recent years. These VQA baselines take the question, the context, and choices as the textual input, take the image as the visual input, and predict the score distribution over choice candidates via a linear classifier. In addition, we build the fine-tuning baseline on top of the large language model UnifiedQA [KMK20]. UnifiedQA takes the textual information as the input and outputs the answer option. Similarly, the image is converted into a caption that provides the visual semantics for the language model.

3.4.3 Language Models with the Chain of Thought

A chain of thought refers to a coherent flow of sentences that reveals the premises and conclusion of a reasoning problem [WWS22b]. A chain of thought clearly decomposes a multi-hop reasoning task into intermediate steps instead of solving the task in a black-box way. The chain of thought can be the step-by-step thought process [WWS22b] before arriving at the final answer or explanations [NRL20] that come after the answer. The annotated lectures and explanations in ScienceQA serve as *demonstrations* of the chain of thought that mimics the multi-step reasoning steps of human beings. In this work, we study if large language models can generate reasonable explanations as the chain of thought to reveal the thought process when answering ScienceQA questions. Further, we explore how the chain of thought can improve the reasoning ability of language models on ScienceQA in both few-shot and fine-tuning learning.

UnifiedQA with the chain of thought. UnifiedQA [KMK20] is a state of the art model for multi-option question answering. The original architecture of UnifiedQA takes the question and options as the input and outputs a short phrase as the final answer. We make a format modification to develop UnifiedQA with the chain of thought (CoT), i.e., UnifiedQA is fine-tuned to generate a

Question: `question` : I_i^{ques}
Options: (A) `option` : I_{i1}^{opt} (B) `option` : I_{i2}^{opt} (C) `option` : I_{i3}^{opt}
Context: `context` : I_i^{cont}
Answer: The answer is `answer` : I_i^{a}. BECAUSE: `lecture` : I_i^{lect} `explanation` : I_i^{exp}

Question: `question` : I_t^{ques}
Options: (A) `option` : I_{t1}^{opt} (B) `option` : I_{t2}^{opt} (C) `option` : I_{t3}^{opt} (D) `option` : I_{t4}^{opt}
Context: `context` : I_t^{cont}
Answer:

Figure 3.11: Prompt instruction encoding for the test example t in GPT-3 (CoT).

long sequence of text which consists of the answer followed by the lecture and explanation.

GPT-3 via chain-of-thought prompting. Recent research work [BMR20, MFL22, LQC23] has shown that GPT-3 [BMR20] can perform various tasks when provided with in-context examples in a standard prompt. Take multi-option question answering as an example, the standard prompt [LBM21, ZWF21, LSZ22] builds instructions using in-context examples with components of the question text, options, and the correct answer text. This style of few-shot learning enables the GPT-3 model to answer specific questions without parameter updates. Different from standard prompting, we build GPT-3 via chain-of-thought (CoT) prompting, as shown in Figure 3.11. To be specific, for each test problem t, we map the prompt instruction $I : \{I_i\}_n, I_t$ into a textual format where $\{I_i\}_n$ refers to the instruction set of n-shot in-context examples from the training set, while I_t denotes the test instruction. Instead of the way where the explanation comes before the answer [WWS22b], we feed the instruction I into the encoder-decoder model GPT-3 to generate the answer a followed by the lecture $lect$ and explanation exp: $M : \{I_i\}_n, I_t \rightarrow a, lect, exp$.

3.4.4 Experimental Setup

Evaluation metrics. The heuristics and VQA baselines treat our ScienceQA task as a multi-class classification problem with multiple options and are evaluated with the accuracy metrics. UnifiedQA and GPT-3 treat ScienceQA as a text generation problem. So the most similar option is selected as the final prediction to evaluate the question answering accuracy. The generated lectures and explanations are evaluated by automatic metrics [PRW02, Lin04, RG19] and human scores by annotators.

Implementation details. The VQA baselines are trained for a maximum number of 50 epochs with a learning rate of $5e{-}5$. We fine-tune the UnifiedQA for $50k$ iterations and evaluate every $1k$ iteration. The training process is stopped following the early stopping strategy with a patience period of three evaluations. For GPT-3, we use the `text-davinci-002` engine, which is the most capable model version suggested in the official documentation. Below are details on the experiments:

- **Fine-tuning on the dataset.** Fine-tuning baselines (VQA baselines and UnifiedQA) are trained on the training set, developed on the validation set, and evaluated on the test set.
- **Input sizes:** For VQA baselines, we set the maximum number of input words or tokens as 100.
- **Batch sizes.** We use batches of 64 and 4 for VQA baselines and fine-tuned UnifiedQA, respectively.
- **Newline character.** For language models, the newline separators (`\n`) in the text are replaced with `\\n` when encoding the inputs because `\n` is normally used as a stop symbol, following the original works [BMR20, KMK20].
- **Captioning model.** We use the tool[1] to generate captions for the images in the dataset. The maximum length of generated captions is 16, the number of beams is 4, and the maximum number of output tokens is 512.
- **Compute resources.** We use two GeForce RTX 3090 GPUs for fine-tuning VQA baselines and UnifiedQA on the dataset.
- **Questions without any context.** For questions without any context, the context text is replaced with an empty string.
- **GPT-3:** Following default settings, we choose temperature, frequency penalty and presence penalty as 0.0, and top probability as 1.0. All experiments for GPT-3 are run via the online API. Experiments in Figure 3.15 are repeated four times.

3.4.5 Results for Question Answering

Table 3.4 demonstrates the empirical results for Science Question Answering.

---

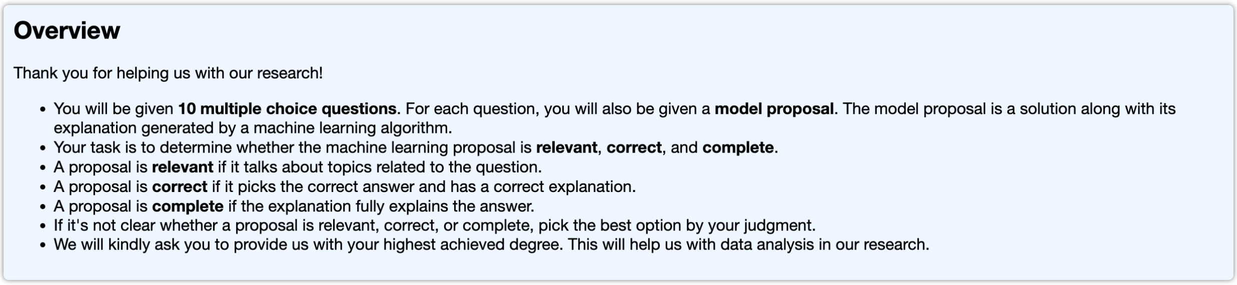
[1] `https://huggingface.co/nlpconnect/vit-gpt2-image-captioning`

Model	**Learning**	**Format**	**NAT**	**SOC**	**LAN**	**TXT**	**IMG**	**G1-6**	**G7-12**	**Avg**
Random chance	-	M→A	40.28	46.13	29.25	47.45	40.08	39.35	40.67	39.83
Q only [AHB18]	train set	Q→A	41.34	27.22	47.00	41.79	35.15	9.28	40.87	39.85
C_I only [AHB18]	train set	C_I→A	41.34	29.25	45.45	42.33	36.09	39.21	41.07	39.87
Q+M only [AHB18]	train set	QM→A	52.66	51.86	60.18	55.57	50.37	52.53	57.88	54.44
Q+C_T+M only [AHB18]	train set	QC_TM→A	57.28	49.04	61.36	60.46	52.80	54.44	60.51	56.61
Q+C_I+M only [AHB18]	train set	QC_IM→A	58.97	53.77	60.45	62.85	54.49	56.72	61.04	58.26
MCAN [YYC19]	train set	QCM→A	56.08	46.23	58.09	59.43	51.17	51.65	59.72	54.54
Top-Down [AHB18]	train set	QCM→A	59.50	54.33	61.82	62.90	54.88	57.27	62.16	59.02
BAN [KJZ18]	train set	QCM→A	60.88	46.57	66.64	62.61	52.60	56.83	63.94	59.37
DFAF [GJY19]	train set	QCM→A	64.03	48.82	63.55	65.88	54.49	57.12	67.17	60.72
ViLT [KSK21]	train set	QCM→A	60.48	63.89	60.27	63.20	61.38	60.72	61.90	61.14
Patch-TRM [LQC21]	train set	QCM→A	65.19	46.79	65.55	66.96	55.28	58.04	67.50	61.42
VisualBERT [LYY19, LYY20]	train set	QCM→A	59.33	69.18	61.18	62.71	62.17	62.96	59.92	61.87
UnifiedQA$_{\text{SMALL}}$ [RSR20]	zero-shot	QCM→A	47.78	40.49	46.00	50.24	44.12	45.56	46.21	45.79
UnifiedQA$_{\text{BASE}}$ [RSR20]	zero-shot	QCM→A	50.13	44.54	48.18	53.08	48.09	47.58	50.03	48.46
UnifiedQA$_{\text{SMALL}}$ [RSR20]	train set	QCM→A	53.77	58.04	61.09	52.10	51.51	58.22	53.59	56.57
UnifiedQA$_{\text{BASE}}$ [RSR20]	train set	QCM→A	68.16	69.18	74.91	63.78	61.38	72.98	65.00	70.12
UnifiedQA$_{\text{BASE}}$ (CoT)	train set	QCM→AE	70.60	74.02	78.36	65.69	64.80	75.48	69.48	$73.33_{3.21\uparrow}$
UnifiedQA$_{\text{BASE}}$ (CoT)	train set	QCM→ALE	71.00	76.04	78.91	66.42	66.53	77.06	68.82	$74.11_{3.99\uparrow}$
GPT-3 [BMR20]	zero-shot	QCM→A	75.04	66.59	78.00	74.24	65.74	76.36	**69.87**	74.04
GPT-3 [BMR20]	2-shot	QCM→A	74.64	69.74	76.00	74.44	67.28	76.80	68.89	73.97
GPT-3 (CoT)	2-shot	QCM→AE	**76.60**	65.92	77.55	**75.51**	66.09	**78.49**	67.63	$74.61_{0.64\uparrow}$
GPT-3 (CoT)	2-shot	QCM→ALE	75.44	**70.87**	**78.09**	74.68	**67.43**	78.23	69.68	$\mathbf{75.17}_{1.20\uparrow}$
Human	-	QCM→A	90.23	84.97	87.48	89.60	87.50	91.59	82.42	88.40

Table 3.4: Evaluation of baselines over different classes in accuracy (%). Model names: Q = question, M = multiple options, C = context, C_T = text context, C_I = image context, CoT = chain of thought. Format names: A = answer, AE = answer with explanation, ALE = answer with lecture and explanation. Question classes: NAT = natural science, SOC = social science, LAN = language science, TXT = text context, IMG = image context, G1-6 = grades 1-6, G7-12 = grades 7-12.

VQA baselines. We feed the VQA baseline models with the input of QCM format to predict answers A. Out of all the VQA models we benchmarked, VisualBERT [LYY19, LYY20] performs the best on average (61.87%). Interestingly, Patch-TRM [LQC21] beats VisualBERT in natural science (NAT) and language science (LAN), and it also performs better in higher-grade questions (67.50% *v.s.* 59.92%). However, in the subject of social science (SOC), VisualBERT outperforms Patch-TRM by a large margin (+22.39%). Such drastic changes in performance might imply that current VQA models are not generalized to process the challenging questions in ScienceQA.

Language models. We evaluate whether large-scale pretraining on text can help language models learn scientific knowledge and thus perform better on the ScienceQA task. For this purpose, we have tried two of the state-of-the-art pre-trained language models: UnifiedQA and GPT-3.



Overview

Thank you for helping us with our research!

- You will be given **10 multiple choice questions**. For each question, you will also be given a **model proposal**. The model proposal is a solution along with its explanation generated by a machine learning algorithm.
- Your task is to determine whether the machine learning proposal is **relevant**, **correct**, and **complete**.
- A proposal is **relevant** if it talks about topics related to the question.
- A proposal is **correct** if it picks the correct answer and has a correct explanation.
- A proposal is **complete** if the explanation fully explains the answer.
- If it's not clear whether a proposal is relevant, correct, or complete, pick the best option by your judgment.
- We will kindly ask you to provide us with your highest achieved degree. This will help us with data analysis in our research.

Figure 3.12: Interface of instructions for AMT workers to evaluate the explanations generated from UnifiedQA (CoT) and GPT-3 (CoT).

(i) **UnifiedQA.** The results show that without any supervised fine-tuning (zero-shot), UnifiedQA cannot beat any VQA baseline model, while the pretraining does help the model obtain some scientific knowledge to outperform the random baseline. When fine-tuned with the answer labels in ScienceQA, UnifiedQA$_{\text{BASE}}$ reports an accuracy of 70.12% on average. By further teaching the model to generate the answer along with lecture and explanation, the developed language model with chain-of-thought (UnifiedQA$_{\text{BASE}}$ (CoT)) brings additional improvements of +3.21% (QCM→AE) and +3.99% (QCM→ALE). These results show that generating the chain of thought along with the answer benefits the reasoning ability of language models.

(ii) **GPT-3.** The positive effect of pretraining is also proved by the surprisingly good results from GPT-3 in the same zero-shot setting as UnifiedQA. Without any fine-tuning, GPT-3 already reaches almost the best performance we can get. Interestingly, prompting the GPT-3 with two training examples with only answers results in a negligible difference. However, if we prompt GPT-3 with chain-of-thought prompting (QCM→ALE), we obtain the state-of-the-art result so far (75.17%).

Human performance. Humans outperform all benchmarks consistently across question classes, context types, and grades, *e.g.,* a 20.07% gap for questions with the image context (IMG) between humans and our best performing model. The gap is to be filled by future research on multimodal reasoning for scientific question answering.

Question: What do these two changes have in common?\n
bleaching clothes\n
a piece of apple turning brown\n
Options: (A) Both are only physical changes. (B) Both are caused by cooling. (C) Both are chemical changes. (D) Both are caused by heating.\n
Answer: **The answer is C. BECAUSE:** A **physical change** is a change to a substance that does not change what the substance is. A **physical change** only changes the substance's physical properties, such as its size, shape, or color.\n
A **chemical change** is a change to a substance that changes what the substance is. A **chemical change** changes the substance's chemical properties, such as its chemical structure.\n
The changes that you listed are both chemical changes.\n
When you **bleach clothes**, the chemical structure of the fabric changes. This change makes the fabric less likely to hold color.\n
When **a piece of fruit turns brown**, the chemical structure of the fruit changes. This change makes the fruit taste different.

Figure 3.13: One example of the predicted answer along with the chain of thought from GPT-3 (CoT).

3.4.6 Human Evaluation of Generated Explanations

We also evaluated the quality of predictions from GPT-3 (CoT) and UnifiedQA (CoT) by asking AMT workers to rate the model-generated explanations. The interface is shown in Figure 3.12. Each sample's question text, contexts, choices, and answers were presented, along with the corresponding explanation generated by language models. The workers were asked to decide whether the proposed explanation is *relevant* (is related to the question), *correct* (gives a correct answer and explanation), and *complete* (fully explains the answer). Prediction outputs that contain textual explanations were grouped into batches of 10, each assigned to 3 workers for evaluation. For each batch, we provided the workers with a monetary compensation of \$0.3. Finally, the human scores for each explanation were determined by taking a majority vote.

One prediction example of GPT-3 (CoT) is visualized in Figure 3.13. We can see that GPT-3 (CoT) predicts the correct answer and generates a reasonable lecture and explanation to mimic the human thought process. We further report automatic metrics (BLEU-1/4 [PRW02], ROUGE-L [PRW02], and (sentence) Similarity [RG19] to evaluate the generated lectures and explanations, as shown in Table 3.5. The Similarity metric computes the cosine-similarity of semantic embeddings between two sentences based on the Sentence-BERT network [RG19]. The results show that UnifiedQA$_{\text{BASE}}$ (CoT) generates the most similar explanations to the given ones. However, it's commonly agreed that automatic evaluation of generated texts only provides a partial view and has to be complemented by a human study. By asking annotators to rate the relevance, correctness, and completeness of generated explanations, we find that the explanations generated by GPT-3 (CoT) conform best to human judgment.

Model	Format	BLEU-1	BLEU-4	ROUGE-L	Similarity	Relevant	Correct	Complete	Gold
UnifiedQA$_{\text{BASE}}$ (CoT)	QCM→ALE	**0.397**	**0.370**	**0.714**	**0.811**	80.4%	76.6%	76.1%	56.9%
GPT-3 (CoT)	QCM→AE	0.234	0.048	0.351	0.561	76.9%	73.0%	70.5%	52.5%
GPT-3 (CoT)	QCM→ALE	0.192	0.052	0.323	0.595	**88.5%**	**78.8%**	**84.5%**	**65.2%**

Table 3.5: Automatic metrics (BLEU-1/4, ROUGE-L, Similarity) and human evaluation of generated explanations. A gold explanation refers to one that is relevant, correct, and complete.

3.4.7 Analysis

Blind studies. Blind studies are conducted on top of the modification of the full model, Top-Down [AHB18]. The results achieved in blind studies of Q only and C_I only are close to random chance, showing that the ScienceQA dataset is robust and reliable in distribution. The performance drops in Q+M only, Q+C_T+M only, and Q+C_I+M only indicate that all input components provide critical information for answering ScienceQA questions.

Prompt types. We study the effect of prompt types and visualize the comparison in Figure 3.14. It shows that prompting the GPT-3 model with both lectures and explanations (QCM→ALE) results in the highest accuracy on average and the smallest variance. In contrast, prompting with only explanations (QCM→AE) gives the largest variance, resulting in a less stable model.

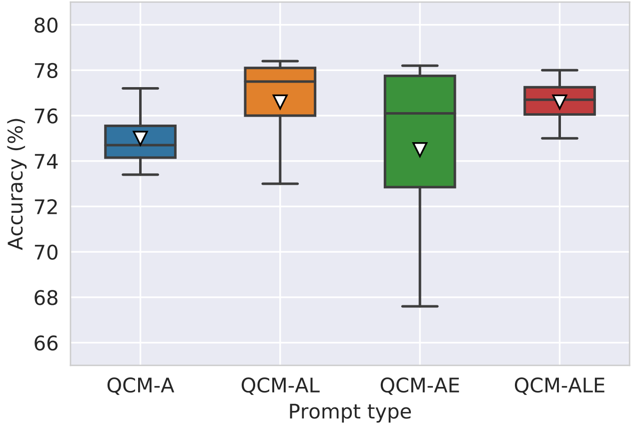


Figure 3.14: Accuracy of GPT-3 (CoT) cross different prompt types with 4-shot examples.

Number of in-context examples. In Figure 3.15, we further investigate how different numbers of training examples encoded in prompts can affect the prediction accuracy. The QCM→ALE prompt type outperforms or performs comparably to the QCM→A type with all numbers of examples. And we observe the peak performance of QCM→ALE with 2 training examples being prompted. After that, the accuracy goes down as more training examples are added to the model.

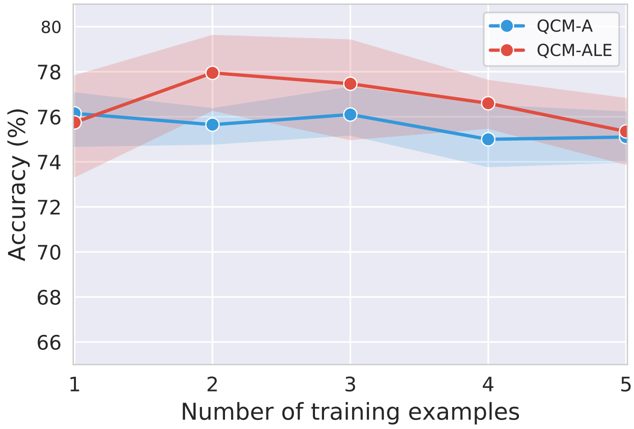


Figure 3.15: Accuracy of GPT-3 (CoT) cross different numbers of training examples (b).

Dynamic sampling. In Table 3.6, instead of random sampling, we try to dynamically select the in-context examples to prompt with the same class as the test sample. However, slight differences in prediction accuracy are observed when comparing them to simple random sampling.

Prompt type	Sampling	Acc. (%)
QCM→ALE	Dynamic (same topic)	75.15
QCM→ALE	Dynamic (same category)	74.58
QCM→ALE	Dynamic (same skill)	75.10

Table 3.6: Dynamic sampling for GPT-3 (CoT).

Upper bound. We search the upper bound of the GPT-3 accuracy by feeding the gold lecture and explanation in the test prompt. As reported in Table 3.7, QCME*→A outperforms the QCM→ALE baseline by 18.86% and QCMLE*→A outperforms QCM→ALE by 18.96%, indicating a potential improvement direction by generating correct explanations before answering science questions.

Positions of lectures and explanations. We study the performance of GPT-3 (CoT) in terms of different positions of lectures and explanations on 1,000 test examples. The results are shown in Table 3.8. There could be huge accuracy decreases if GPT-3 (CoT) predicts lectures and explanations before answers. It is mainly because if GPT-3 (CoT) is formulated to generate the long

Prompt type	**Sampling**	**Acc. (%)**
QCML*→A	Random	73.59
QCML*→AE	Random	74.32
QCME*→A	Random	94.03$_{18.86\uparrow}$
QCMLE*→A	Random	**94.13**$_{18.96\uparrow}$
QCM→ALE	Random	75.17

Table 3.7: Upper bound of GPT-3 (CoT).

lecture and explanation first, there is a greater chance that it will stop generating the prediction early or use up the maximum token limits before obtaining the required answer.

Prompt type	**Sampling**	**Acc. (%)**
QCM→LA	Random	60.6
QCM→EA	Random	56.0
QCM→LEA	Random	55.4
QCM→ELA	Random	51.5
QCM→ALE	Random	**73.6**

Table 3.8: Different positions of L/E for GPT-3 (CoT).

CoT learns with fewer data. To study if the chain of thought helps language models learn more efficiently, we report the accuracies of UnifiedQA and UnifiedQA (CoT) fine-tuned on different sizes of the training set in Figure 3.16. UnifiedQA (CoT) benefits language models by learning the coherent reasoning path when answering questions, resulting in similar accuracy with fewer training examples.

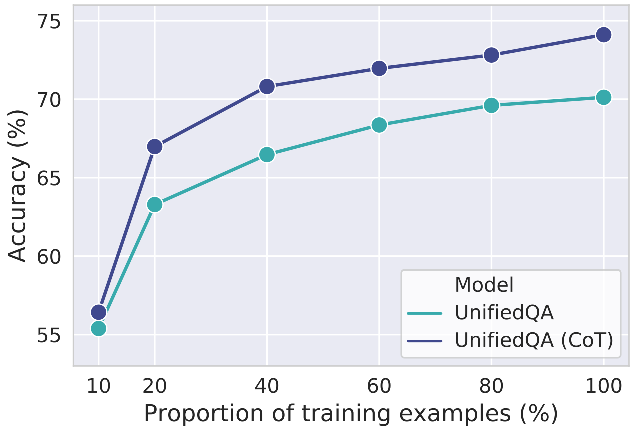


Figure 3.16: UnifiedQA (CoT) learns efficiently with fewer training examples.

3.4.8 Case Study

Successful cases. We visualize three examples with correct answers and gold explanations predicted by GPT-3 via *chain-of-thought* prompting (CoT) in Figures 3.17 to 3.19. We can see that GPT-3 (CoT) not only predicts the correct answers but also generates reasonable explanations, which follow the multi-hop reasoning process of human beings. This suggests that large language models like GPT-3 have great promise for implementing high-level reasoning abilities.

Question: Which animal's neck is also adapted for hunting prey while keeping the rest of its body still?
Context: Saddle-billed storks live near wetlands and lakes. They eat mostly fish. The 's neck helps it grab fish while keeping the rest of its body still. If the stork had to move its body, it might scare the fish away.
Figure: saddle-billed stork. A black and white bird is standing in the grass.
Options: (A) northern pintail (B) black-headed heron
Answer: **The answer is B. BECAUSE:** The **northern pintail** is a duck. It does not hunt for fish.`\n`
The **black-headed heron** is a type of heron. It lives near wetlands and lakes. It eats mostly fish.`\n`
The heron's neck helps it grab fish while keeping the rest of its body still. If the heron had to move its body, it might scare the fish away.

(**Subject:** Natural science, **Topic:** Biology, **Category:** Adaptations, **Skill:** Animal adaptations: beaks, mouths, and necks)

Figure 3.17: A natural science example with the correct answer and a gold explanation.

Question: Suppose Mason decides to bake blueberry muffins. Which result would be a cost?
Context: Mason is deciding whether to bake blueberry muffins or cranberry muffins. He wants the muffins to be tasty. But he also wants to make a lot of muffins.
Options: (A) Mason will get to eat blueberry muffins. He thinks blueberry muffins are tastier than cranberry muffins. (B) Mason will give up some muffins. He could have made more cranberry muffins than blueberry muffins.
Answer: **The answer is B. BECAUSE:** A **cost** is anything that you have to give up to get something else.
In this case, **Mason will have to give up some muffins**. He could have made more cranberry muffins than blueberry muffins.
So, the **cost** of making blueberry muffins is the opportunity **cost** of not making cranberry muffins.

(**Subject:** Social science, **Topic:** economics, **Category:** Basic economic principles, **Skill:** Costs and benefits)

Figure 3.18: A social science example with a correct answer and a gold explanation.

Failure cases. GPT-3 via chain-of-thought prompting obtains promising results but still fails to answer a wide range of challenging questions in ScienceQA. The failure cases can be classified into two types: (a) the model fails to understand the multimodal inputs and lacks domain-specific knowledge to arrive at the correct answer; (b) the model generates the wrong chain of thought with irrelevant, incorrect, or incomplete information.

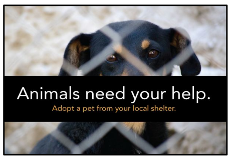
Figures 3.20 to 3.22 visualize three examples with predictions from GPT-3 (CoT). In these examples, GPT-3 (CoT) is able to predict the correct answers but fails to generate gold explanations.

Question: Which sentence uses a metaphor?
Context: N/A
Options: (A) The cat's silver eyes were two shiny coins. (B) The cat's silver eyes were like two shiny coins.
Answer: **The answer is A. BECAUSE:** A **metaphor** is a figure of speech that uses one thing to describe another. A **metaphor** is a type of analogy.\n
In a **metaphor**, one thing is said to be another thing.\n
For example, you might say, "My dad is a bear."\n
This means that your dad is big and strong, not that he is actually a bear!\n
The sentence "The cat's silver eyes were two shiny coins" is a metaphor.\n
The sentence "The cat's silver eyes were like two shiny coins" is **not a metaphor**. It is a **simile**.\n
A **simile** is a figure of speech that uses the words "like" or "as" to compare two things.

(**Subject:** Language science, **Topic:** Figurative language, **Category:** Literary devices, **Skill:** Identify similes and metaphors)

Figure 3.19: A language science example with a correct answer and a gold explanation.

Question: Which is the main persuasive appeal used in this ad?
Context: A black dog standing in front of a fence.
Options: (A) logos (reason) (B) ethos (character) (C) pathos (emotion)



Answer: **The answer is C. BECAUSE:** The ad is trying to make you feel a certain way about the product. It is not trying to persuade you with logic or with an appeal to character.

(**Subject:** Language science, **Topic:** Writing strategies, **Category:** Persuasive strategies, **Skill:** Identify appeals to ethos, pathos, and logos in advertisements)

Figure 3.20: An example with a correct answer but an irrelevant explanation.

For example, GPT-3 (CoT) generates an *irrelevant* explanation because the context text does not include fine-grained visual information in the image (Figure 3.20). In the example shown in Figure 3.21, GPT-3 (CoT) fails to predict coherent thought chains, providing an *incorrect* example and an *incorrect* statement for a *chemical change*. The third example is given in Figure 3.22, where the generated explanation is just a repetition of the input question and the output answer, instead of following the *complete* thought chain to arrive at the final answer.

Three failure examples with wrong predicted answers are listed in Figures 3.23 to 3.25. We extract the image captions and feed them to the large language model as the visual content input. However, these captions lack fine-grained semantics and usually do not work well for diagrams, which results in two failure cases shown in Figure 3.23 and 3.24. Moreover, there exist challenges for large language models to reason about the questions that require them to understand complex and uncommon domain knowledge. For example, GPT-3 (CoT) cannot understand accurately the terminology of *personification* in language science (Figure 3.25).

Question: What do these two changes have in common?
mixing sand and gravel
crushing a mineral into powder
Context: N/A
Options: (A) Both are chemical changes. (B) Both are only physical changes. (C) Both are caused by cooling. (D) Both are caused by heating.
Answer: **The answer is B. BECAUSE:** A **physical change** is a change to a substance that does not change what the substance is. A **physical change** only changes the substance's physical properties.\n
For example, when you cut a mineral, you change its shape. But the mineral is still the same substance. So, cutting a mineral is a **physical change**.\n
A **chemical change** is a change to a substance that changes what the substance is. A **chemical change** changes the substance's chemical properties.\n
For example, when you mix sand and gravel, you make concrete. Concrete is a new substance with different properties from sand and gravel. So, mixing sand and gravel is a **chemical change**.\n
Crushing a mineral into powder is a **physical change**. The mineral is still the same substance.

(**Subject:** Natural science, **Topic:** Chemistry, **Category:** Physical and chemical change, **Skill:** Compare physical and chemical changes)

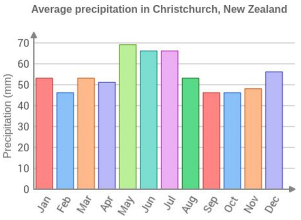
Figure 3.21: An example with a correct answer but an incorrect explanation.

Question: Is the following trait inherited or acquired?
Marshall has five fingers on each hand.
Context: Hint: Marshall was born with five fingers on each hand.
Options: (A) acquired (B) inherited
Answer: **The answer is B. BECAUSE:** Marshall was born with five fingers on each hand. This trait was inherited from Marshall's parents.

(**Subject:** Natural science, **Topic:** Biology, **Category:** Heredity, **Skill:** Identify inherited and acquired traits)

Figure 3.22: An example with a correct answer but an incomplete explanation.

Question: Which month is the wettest on average in Christchurch?
Context: Use the graph to answer the question below. A series of photographs showing a variety of colorful objects.
Options: (A) April (B) December (C) May



Answer: **The answer is B. BECAUSE:** The graph shows the average monthly rainfall in Christchurch from 1971 to 2000.\n
The wettest month on average is December, with an average rainfall of 154.8 mm.\n
The second wettest month is May, with an average rainfall of 153.1 mm.\n
The third wettest month is April, with an average rainfall of 151.4 mm.

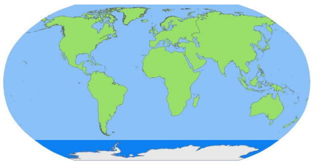
(**Subject:** Natural science, **Topic:** Earth science, **Category:** Weather and climate, **Skill:** Use climate data to make predictions)

Figure 3.23: An example with a wrong answer and a wrong explanation.

3.5 Conclusion

We propose ScienceQA, a dataset that features 21,208 multi-option questions with multimodal contexts from the science curriculum. To the best of our knowledge, ScienceQA is the first large-

Question: Which ocean is highlighted?
Context: A painting of a penguin on a blue background.
Options: (A) the Indian Ocean (B) the Arctic Ocean (C) the Pacific Ocean (D) the Southern Ocean



Answer: **The answer is C. BECAUSE:** The Pacific Ocean is the largest ocean. It covers more than one-third of the Earth's surface.

(**Subject:** Social science, **Topic:** Geography, **Category:** Physical Geography, **Skill:** Oceans and continents)

Figure 3.24: An example with a wrong answer and a wrong explanation.

Question: Complete the sentence so that it uses personification.
At night, when the machines were () the factory seemed to be an entirely different place.
Context: N/A
Options: (A) off (B) asleep
Answer: **The answer is A. BECAUSE:** **Personification** is a figure of speech that gives human qualities to nonhuman things.\n
For example, the sentence "The wind was howling" uses **personification**. It gives the human quality of sound to the nonhuman thing wind.\n
The sentence "The sun was smiling" uses **personification**. It gives the human quality of expression to the nonhuman thing sun.\n
In the sentence "At night, when the machines were **off**, the factory seemed to be an entirely different place," the machines are given the human quality of sound.

(**Subject:** Language science, **Topic:** Writing strategies, **Category:** Creative techniques, **Skill:** Use personification)

Figure 3.25: An example with a wrong answer and a wrong explanation.

scale multimodal science dataset where most questions are annotated with corresponding lectures and explanations. We establish various baselines, including recent VQA models and large language models on ScienceQA. We further study if language models can generate reasonable explanations and then benefit the reasoning ability. Experiments show that UnifiedQA with the chain of thought can achieve an improvement of 3.99% and few-shot GPT-3 via chain-of-thought (CoT) prompting can obtain a satisfactory accuracy of 75.17% on ScienceQA. 65.2% of the generated explanations from GPT-3 (CoT) meet the gold standard by human evaluations.

Part II

Pre-trained Vision-Language Models

CHAPTER 4

Patch Cross-modal Transformer Model

4.1 Introduction

We have witnessed the exciting development of visual question answering (VQA) research in recent years. The long-standing goal of the VQA task is to develop systems that can answer natural questions that correspond to visual information. Several datasets have been released to evaluate the systems' visual and textual content understanding abilities [AAL15, ZGB16, GKS17, JHM17, HM19, WLS20]. Besides, numerous methods are proposed to push the limit of performing visual reasoning over natural images [AHB18, KJZ18, YYC19, GJY19].

One of the underlying limitations of current VQA works is that they are focusing on answering visual questions for natural images. However, besides natural images, abstract diagrams that carry visual and semantic richness, account for a large part of the visual world. For instance, it is shown that emojis can express rich human sentiments [KSK16, FMS17], and diagrams like icons can map the physical worlds into symbolic and aesthetic representations [LGG19, MBT18, KBY20]. Some pioneering works attempt to design systems that are capable of answering questions for abstract diagrams. However, these benchmarks either address domain-specific charts, plots and, illustrations [KSK16, KPC18], or are generated from limited templates [ZGS16, SLY17, JHM17]. These limitations impede their practical applications in real-world scenarios, such as elementary education, where abstract diagrams are involved with diverse objects and various reasoning skills [Kar11].

To address these shortcomings, we have introduced icon question answering (IconQA), a new task for *abstract* visual reasoning and compositional question answering in Section 2.2. The task, stemming from the math word problems [Mar08] for children, exhibits a promising potential to

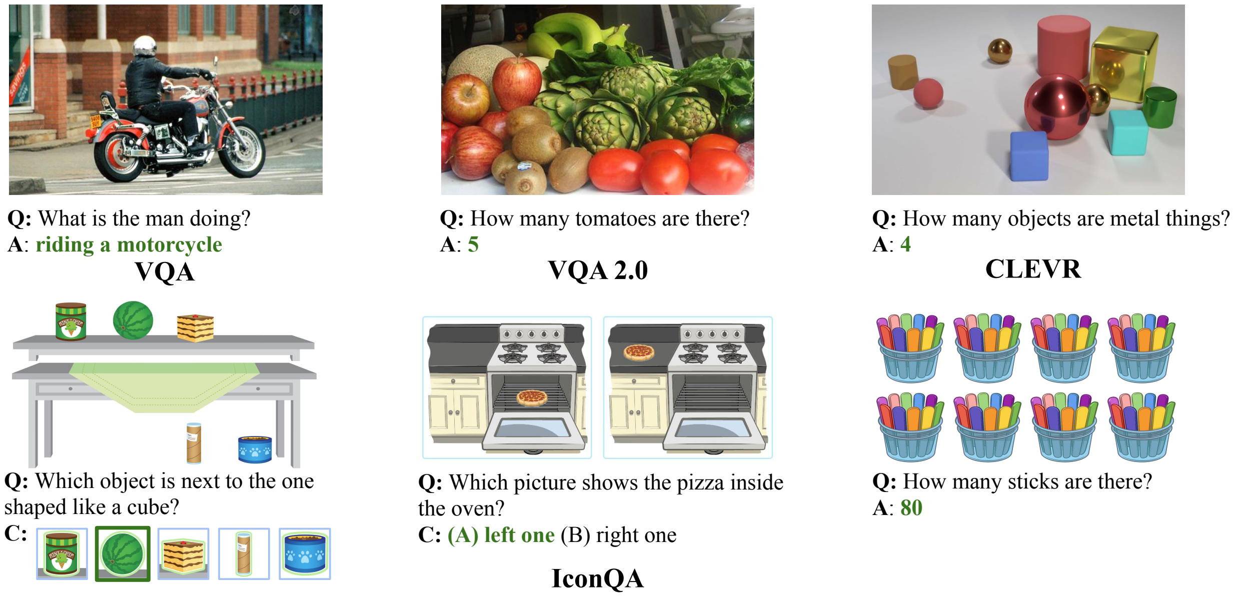

Q: What is the man doing?
A: riding a motorcycle
VQA
Q: How many tomatoes are there?
A: 5
VQA 2.0
Q: How many objects are metal things?
A: 4
CLEVR
Q: Which object is next to the one shaped like a cube?
C:
Q: Which picture shows the pizza inside the oven?
C: (A) left one (B) right one
Q: How many sticks are there?
A: 80
IconQA

Figure 4.1: Top: Examples in three popular VQA datasets: VQA [GKS17], VQA 2.0 [GKS17], and CLEVR [JHM17]. Bottom: Examples of three sub-tasks in our collected IconQA dataset.

develop education assistants. We name the proposed task as IconQA because the images depict icons, which simplify recognition and allow us to focus on reasoning skills for further research. A typical IconQA problem is provided with an icon image and a question, and required to give the answer or choose one from multiple choices in text or image formats.

Correctly answering IconQA questions needs diverse human intelligence skills. As examples shown in Figure 4.1, it requires perceptual abilities such as understanding text and abstract diagrams, including recognizing objects and identifying attributes. It further requires cognitive skills like counting objects, comparing attributes, performing arithmetic operations, making logical inference, completing spatial reasoning, or leveraging external commonsense. Thus, these inherent complexities of IconQA make it a challenging task.

Current paradigms [AHB18, KJZ18, GJY19] for vision-language tasks typically depend on multimodal fusion approaches between two modalities (like single-hop attention over image regions conditioned on the input question), which limit the types of potential interactions within modalities. Inspired by recent advances that Transformer has achieved for achieving fine-grained interactions for input tokens, we propose a novel pyramid patch cross-modal Transformer (Patch-TRM) to learn implicit visual and linguistic relationships in IconQA (Figure 4.2). Patch-TRM fuses the diagram and question inputs into a common semantic embedding space within a multi-modal Transformer, which applies self-attention and cross-modal attention mechanisms to learn intra- and inter-modality interactions homogeneously.

Existing Transformer-based methods [LBP19, LYY19] need pre-extracted object region proposals learned from natural images, leading to the domain gap when processing icon images in IconQA. Given that the icon objects are arranged in grid-like regions with varied sizes, we parse the diagrams into patch sequences in a spatial pyramid structure. Then patches are encoded by a ResNet network [HZR16] and fed to transformer encoders for outputting patch embedding by learning their hierarchical relationships. To enhance the semantic representations for the diagrams, we use IconQA, a large-scale icon dataset, to pretrain the backbone ResNet in the icon classification task. Experiments show that our proposed method is capable of extracting meaningful visual representations for icon images and thus achieving better performances than current advanced VQA methods in the IconQA task.

Our contributions can be summarized as 1) we propose a new task IconQA that requires abstract diagram understanding for icon images and diverse visual reasoning skills; 2) unlike existing methods limited on natural images, our novel cross-modal Transformer model with the pretraing technology can extract hierarchical and informative visual representations for icon images and then significantly outperforms previous work for the IconQA task.

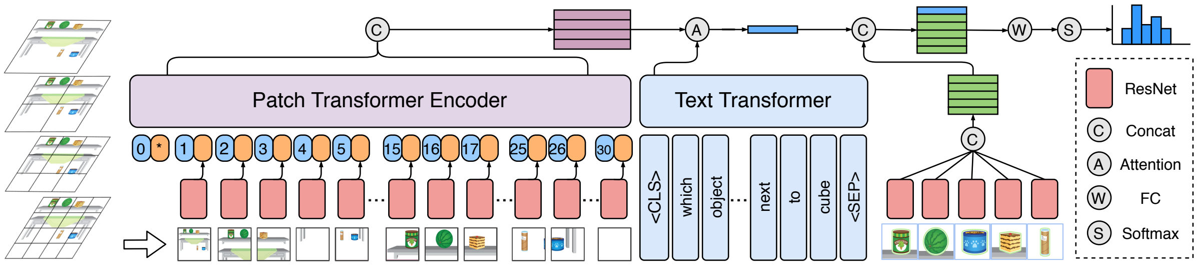
4.2 Related Work

VQA Methods. Early VQA approaches usually combine multi-modal inputs by applying attention mechanisms over image regions or question words [KJZ18, LLZ18, LJZ18, GLL18, YYC19, GJY19]. Inspired by the semantic nature of VQA images, a line of approaches adopt object proposals from pre-trained object detectors and learn their semantic relationships [KJZ18, YYC19, GJY19]. As Transformers achieve excellent performance on vision tasks, pioneering works have attempted to use pre-trained models to learn visual representations for natural images in the VQA task [LBP19, LYY19, CLY20, KSK21] and achieve significant improvements. However, current VQA models are not capable of extracting meaningful visual representations from abstract diagrams, as they require image embeddings or object proposals learned from natural images. Instead, we develop a strong baseline that feeds spatial patch sequences into a Transformer encoder that is powered by the embedding module pre-trained on our Icon645 dataset.

Pretrained Models. After the transformer model achieves excellent performance on translation tasks, large-scale pre-trained model BERT [KT19] and its variants such as TinyBERT [JYS20] have become dominant language models. Inspired by these achievements, we apply the BERT model pre-trained on a large-scale corpus to encode questions in IconQA. When it comes to the vision side, progress is lagging behind for pre-trained models. Some pioneering works have attempted to use the pre-trained models to learn visual representations for natural images in vision-language tasks like VQA [LBP19, LYY19, CLY20] and achieve significant improvements. However, current visual Transformer models for VQA are not capable of extracting meaningful visual representations for abstract diagrams, as they require to extract object proposals from object detectors trained by natural images. Instead, we propose a novel visual model that feeds spatial patch sequences into a transformer encoder and then learns their representations by self-attention.

4.3 Patch Cross-modal Transformers

Inspired by recent advances Transformer has achieved in vision-language tasks [LYY19, LBP19], we develop a cross-modal Transformer model Patch-TRM for icon question answering. Taking the *multi-image choice* sub-task as an example, the overall architecture is shown in Figure 4.2. The diagram is first parsed into ordered patches in a hierarchical pyramid layout. These patches are then encoded by a pre-trained ResNet and passed through a vision Transformer. Question text is encoded by a language Transformer and fused with patch embeddings via the attention mechanism. The encoded image choices are concatenated with the joint diagram-question representation and then fed to a classifier for question answering. The other two sub-tasks utilize similar network architectures, except that in the *multi-text-choice* sub-task, we use an LSTM encoder [HS97] for choice embedding, while *filling-in-the-blank* does not need a choice encoder.



(a) Pyramid Layout (b) Patch Cross-modal Transformer for IconQA

Figure 4.2: Our proposed model Patch-TRM for the IconQA task.

4.3.1 Diagram Encoder

Similar to natural images in most VQA datasets, abstract diagrams also have rich visual and semantic information that is critical to answering questions. Current dominant VQA methods [AAL15, AHB18, KJZ18, GJY19, YYC19, JMR20] either extract high-level visual representations from a pre-trained ResNet backbone network [HZR16] in a top-down fashion, or apply a bottom-up mechanism to extract semantic representations via a object detector, such as a model based on Faster R-CNN [RHG15]. However, these methods depend heavily on the backbone network, which is pre-trained on natural images. When processing diagrams in IconQA, they are likely to fail to extract meaningful representations or reasonable object proposals. Inspired by the early progress in using hierarchical scene layout to parse images [LSF10, ZWZ15, WWZ15] and the recent advances in Transformer-based image encoding [LBP19, LYY19, Won21], we develop a method that splits diagrams into hierarchical patch sequences from a pyramid structure and learns their visual representations using a visual Transformer.

As diagrams in IconQA have more varied aspect ratios than natural images, we add blank paddings at the bottom or on the right side of the images to ensure that they are square-shaped. Each padded diagram is then cropped into a set of patch sequences with different scales. The padding operation and the hierarchical scene layout can facilitate extracting complete objects that retain specific semantics. Let $p = [p_1, p_2, \ldots, p_n]$ denote the patch sequence in the splitting order from the original diagram. From each patch sequence, we extract the visual features using a ResNet model and represent the features as $f_p = [f_{p_1}, f_{p_2}, \ldots, f_{p_n}]$. The representation for each patch, f_{p_i}, is then summed up with its positional embedding with respect to its sequencial index i. Finally, the updated visual patch embeddings pass through a standard multi-layer Transformer [VSP17] to learn high-level visual representations $h_p = [h_{[\texttt{CLS}]}, h_{p_1}, h_{p_2}, \ldots, h_{p_n}]$. Here, the trainable token [CLS], which is added to the Transformer inputs, learns the global meaning of these sequences. As mentioned before, it is not feasible to use existing pre-trained ResNet to process abstract diagrams due to a lack of similar resources for pre-training. So we pre-train the ResNet on icon classification with the icon dataset we compiled (Section 4.4.1). More details of the pre-training task are discussed in Section 4.4.2.

4.3.2 Language Encoder

Questions in IconQA have a wide distribution of question lengths, so we follow the recent approaches [VSP17, JYS20, TCL19, LYY19, LBP19] that apply the BERT model [KT19] to embed question texts, rather than using traditional LSTM [HS97] or GRU [CMG14] for long sequence encoding. Given the question $w_0, w_1, \ldots, w_t$, the input is formatted as $[[\texttt{CLS}], w_0, w_1, \ldots, w_t]$. We use the WordPiece [SN12] subword tokenizer and the resulting sequence is padded to the maximum length. Similar to other methods that use BERT as sentence encoders, we consider the output corresponding to the first token [CLS] as the embedding of the entire question, noted as h_q.

4.3.3 Answer Reasoning

Given the image patch representation $h_p \in \mathcal{R}^{n \times k}$, and question embedding $h_q \in \mathcal{R}^k$, where n denotes the number of diagram patches and k denotes the learned embedding size of the patches, we apply a cross-modal attention to learn their joint representation:

$$a = \text{softmax}\left(W_p h_p \circ W_q h_q\right), \tag{4.1}$$

$$h_v = \sum_i^n a(i) \times h_{p_i}, \tag{4.2}$$

where W_p and W_q are learnable mapping parameters, and $\circ$ is the element-wise product operator. The joint representation h_v is calculated as the weighted sum over all diagram patches.

Before predicating the answer, multiple choice candidates need to be encoded. Taking the *multi-image-choice* task as an example, each image choice is encoded as the output of the last pooling layer of the pre-trained ResNet. The encoded image choice is denoted as $h_c \in \mathcal{R}^{m \times k}$, where m is the number of the candidates. The choice embeddings are concatenated with the diagram-question representation, and then the resulted embeddings are fed to a classifier over the candidates:

$$p_{ans} = \mathrm{softmax}\left(W_a\left([h_v; h_c]\right) + b_a\right), \tag{4.3}$$

where W_a and b_a are classifier parameters, and p_{ans} is the probability of the predicated answer choice.

Similarly, in the *multi-text-choice* sub-task, the answer is predicated over text choices, except that each text choice is embedded with LSTM layers first. We formulate the *filling-in-the-blank* sub-task as a multi-class classification problem from all possible answers in the training data, as most VQA works do. After generating the joint encoding for the input diagram and question, a linear classifier is trained to predict the final answer.

4.4 Image Encoder Pre-training

Current dominant VQA methods either rely heavily on the ResNet backbone network to extract image features or depend on the Transformer encoders to learn image embeddings. However, these networks are pre-trained on natural images and are likely to fail to extract meaningful representations or reasonable object proposals when processing the diagrams in IconQA. Instead, we pre-train the ResNet network on the icon classification task with the icon dataset we compiled (Section 4.4.1). Patch-TRM hierarchically parses the diagram into patches that retain complete objects to a large extent, and the parsed patches are embedded by the pre-trained ResNet network before being fed into the vision Transformer. The hierarchical parsing structure, along with the ResNet pre-trained on icon data facilitate our Patch-TRM to learn informative diagram representations for the IconQA task.

Data	**#Classes**	**#Icons**	**Min Size**	**Max Size**	**Colored**
Icon645	377	645,687	64×64	256×256	✓

Table 4.1: Statistics for the Icon645 dataset.

4.4.1 Pre-training Data: Icon645

As discussed in Section 2.2.3, IconQA questions are accompanied by abstract diagrams that cover a wide range of icon objects. Using existing backbone networks to extract image representations for these icon images is inadequate, as most of these networks are pre-trained on natural images. To overcome the limitation, we develop a new large-scale icon dataset for pre-training existing vision backbone networks. We use the collected icon data to pre-train the current backbone networks, which can be applied to extract diagram representations in IconQA.

We retrieve the 388 icon classes mentioned in the question texts from Flaticon[1], the largest database of free vector icons. After removing 11 classes that can't be retrieved, we construct an icon dataset containing 377 classes, called Icon645. As summarized in Table 4.1, the Icon645 dataset includes 645,687 colored icons with a minimum size of 64 by 64 and a maximum size of 256 by 256. Examples in Table 4.2 show that our collected icons include a wide variety of colors, formats and styles. On top of pre-training encoders, the large-scale icon data could also contribute to future research on abstract aesthetics and symbolic visual understanding. In this work, we use the icon data to pre-train backbone networks on the icon classification task in order to extract semantic representations from abstract diagrams in IconQA.

Icons	**Examples**	**Icons**	**Examples**
Bed		Bucket	
Cake		Car	
Castle		Dog	
Giraffe		Kite	
Soda		Tree	

Table 4.2: Collected icon examples in the Icon645 dataset.

---

[1] Flaticon: `https://www.flaticon.com/`

Method	Total	Head	Medium	Tail
ResNet32 [HZR16] + CB [CJL19]	27.91	19.66	36.51	33.53
ResNet32 [HZR16] + Focal Loss [LGG17]	32.80	51.59	36.51	8.94
ResNet32 [HZR16] + LDAM [CWG19]	42.65	55.68	46.42	24.94
ResNet101 [HZR16] + LDAM [CWG19]	**62.93**	**70.29**	**70.50**	**47.51**

Table 4.3: Results for icon classification.

4.4.2 Icon Classification for Pre-training

The Icon645 dataset is collected to pre-train the backbone network for patch feature extraction. The dataset has a long-tailed distribution, and thus we address the class-imbalanced issue following previous studies on specific loss functions such as CB loss [CJL19], Focal loss[LGG17], and LDAM loss [CWG19]. The metric of Top-5 accuracy is used to evaluate different model setups and the evaluation results are summarized in Table 4.3. Following [LMZ19], to demonstrate performances on different data parts, we divide the dataset into three balanced clusters: Head, Medium, and Tail, corresponding to 132, 122, and 123 classes respectively. All classes in Head have at least 1,000 instances, all classes in Medium have 300 - 999 instances, and all classes in Tail have fewer than 300 instances. As the results show, the backbone network ResNet101 with a re-balanced LDAM loss function achieves the best result for icon classification on Icon645. Consequently, we adopt this pre-trained ResNet101 network to extract patch features in our baseline Patch-TRM for IconQA.

4.5 Experiments

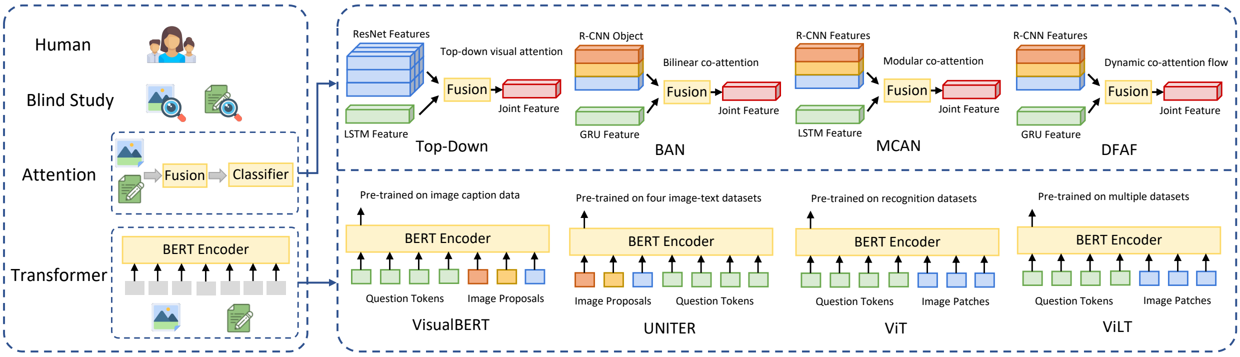


Figure 4.3: An overview of benchmark baselines on the IconQA task.

4.5.1 Baselines

For comparison, we consider multi-modal pooling methods with attention mechanisms [AHB18, KJZ18, GJY19, YYC19], Transformer-based VQA approaches [LBP19, CLY20, Won21, KSK21], and three blind study methods as benchmark models, as summarized in Figure 4.3. Additionally, a user study is conducted to explore the performances of human beings in different age groups.

Attention models. We construct four attention models for benchmarking. The first model implements Top-Down attention [AHB18] for VQA, which is a strong attention method that applies free-form based attention on image representations from a pre-trained ResNet-101 network. The remaining three models utilize the bottom-up attention mechanism with the help of object detection proposals from Faster-RCNN [RHG15]. Specifically, BAN [KJZ18] proposes a method that utilizes bilinear attention distributions to learn joint vision-language information. DFAF [GJY19] is an advanced model that applies self-attention and cross-modal attention and introduces the information flow to help the model focus on target question words and image regions. The last approach, MCAN [YYC19], learns the self-attention on the questions and images and the question-guided-attention of images jointly.

Transformer models. Four Transformer-based models are also implemented as benchmarks. ViLBERT [LBP19] and UNITER [CLY20] are two Transformer-based approaches that take image object proposals from Faster-RCNN [RHG15] and question tokens as inputs. Specifically, ViLBERT learns the joint representation of the image content and the natural language content from image proposal regions and question tokens, while UNITER processes multimodal inputs simultaneously for joint visual and textual understanding. The last two benchmarks ViL [Won21] and ViLT [KSK21] are more recently proposed Transformer models that take image patch tokens instead of object proposals as inputs when representing the image.

Blind study models. We develop three models to check for possible data biases in the IconQA dataset. A random baseline picks up one from the given choice candidates for the *multiple-choice* sub-tasks while predicts the answer by randomly selecting one from all possible answers in the train data for the *filling-in-the-blank* sub-task. Q-Only is set up similar to the Top-Down [AHB18] model, but it only considers textual inputs. This baseline learns the question bias in the training set. I-Only also has a Top-Down architecture, but it only takes abstract diagrams as inputs, and tests the distribution biases in the images and answers in IconQA.

User study. To assess human performances in the IconQA task, we post the test set of IconQA on Amazon Mechanical Turk (AMT) and ask people to provide answers to the questions in the test set. We also ask the participants to provide us with their age group anonymously. We strongly encourage parents who have young children to let their children complete the questionnaires, as their answers give us insights to how the designed audience of these questions perform. Further details about the user study are included in Appendix 4.7.

4.5.2 Experimental Details

Following prior work [AAL15], all the baselines are trained on the IconQA training set, tuned on the validation set, and finally evaluated on the test set. Similar to [AAL15], we choose accuracy as the evaluation metric. For the two *multi-choice* sub-tasks, the answer is regarded as correct only if it matches the ground truth. On the other hand, as the collected answers for *filling-in-blank* are short numbers, correct answers are expanded to include both the digital number and its corresponding words.

All experiments are run on one Nvidia RTX 3090 GPU. We use the Adamax optimizer with optimal learning rates of 7×10^{-4}, 8×10^{-4}, and 2×10^{-3} on the three sub-tasks respectively. We apply a binary cross-entropy loss to train the multi-class classifier with a batch size of 64 and a maximum epoch of 50. The early stopping strategy is used when the validation accuracy stops improving for five consecutive epochs. It takes about 50, 30, and 10 minutes to train our baseline Patch-TRM on three sub-tasks respectively.

We use the same learning parameters set in Top-Down [AHB18] when evaluating the eight baselines listed in Section 4.5.1 and our developed baseline Patch-TRM. Some crucial parameters used in our model are clarified below.

Patch-TRM. For our approach Patch-TRM, each diagram is split four times by varied scales, resulting in 79 (1+4+9+16+49) patches totally. After resizing them to to 224×224, patch visual features are extracted from the last pooling layer, resulting in a 2048-d feature vector. The ResNet network used to embed the patches is pre-trained on the icon classification task as discussed in Section 4.4.2. The patch Transformer has one layer of Transformer block with four attention heads and outputs embeddings with a hidden state size of 768. A small pre-trained BERT model [TCL19] is used to encode the question text in the language encoder.

Attention models. For Top-Down, the attention-based baselines use $7 \times 7 \times 2048$-d features from the last convolution layer. For BAN [KJZ18], DFAF [GJY19], and MCAN [YYC19], image features of dimension 2,048 are extracted from Faster R-CNN [RHG15]. Question words in these attention models are encoded into features of dimension 1,024 by GRU [CMG14]. And the visual and textual features are then embedded into 1,024 dimensions with the corresponding attention mechanisms and fusion methods reported in original works.

Transformer models. For ViLBERT [LBP19] and UNITER [CLY20], we use Faster R-CNN [RHG15] to extract 36 proposal regions as the visual inputs. Both ViL [Won21] and ViLT [KSK21] use ViT-B/32 pre-trained on ImageNet to encode the image emebeddings. The hidden size is set as 768, the layer depth is 32, and the input image is sliced into patches with a size of 32. For ViL, we use two dependent Transformers to embed the question and image respectively.

4.5.3 Experimental Results

Table 4.4 demonstrates the results of the benchmark methods and our baseline on the IconQA test set. The first three columns of the results represent the three sub-tasks: *multi-image-choice*, *multi-text-choice*, and *filling-in-the-blank* respectively. The remaining 13 columns illustrate the results of these approaches over problems that require different reasoning skills, as defined in Table 4.4.

Human performance. Out of the 54,896 collected answers, 9,620 are made by young children from age 3 to 8, 19,040 are made by adolescents from age 9 to 18, and 26,236 are made by adults. The human performance over the three sub-tasks and thirteen skills is illustrated in Figure 4.4. The detailed results for human performance in the IconQA task are shown in Table 4.5. As expected, young children do not answer the questions as well as adolescents or adults, suggesting that most participants answered their ages correctly. Moreover, the results show that humans perform more consistently on all sub-tasks compared to machine algorithms. Interestingly, humans

	Sub-tasks (3)			**Reasoning skills (13)**												
Method	Img.	Txt.	Blank	Geo.	Cou.	Com.	Spa.	Sce.	Pat.	Tim.	Fra.	Est.	Alg.	Mea.	Sen.	Pro.
Human	95.69	93.91	93.56	94.63	97.63	94.41	93.31	92.73	95.66	97.94	97.45	87.51	96.29	86.55	97.06	85.67
Random	41.70	36.87	0.29	30.30	18.38	41.20	36.49	34.25	34.81	35.82	34.84	3.62	11.12	0.36	45.16	38.81
Q-Only	41.64	36.86	28.45	38.03	33.63	48.19	37.14	35.37	33.66	48.09	33.06	40.46	28.02	38.07	45.25	40.76
I-Only	41.56	36.02	46.65	38.71	37.64	45.26	37.52	35.47	36.29	47.37	32.48	62.29	31.73	64.02	45.25	37.51
Top-Down [AHB18]	75.92	68.51	73.03	80.07	65.01	80.65	45.78	58.22	55.01	68.28	72.43	**99.54**	50.00	**99.46**	84.54	83.75
BAN [KJZ18]	76.33	70.82	75.54	79.99	67.56	82.12	53.20	66.92	55.67	66.50	73.77	97.06	47.46	96.50	82.12	82.45
ViLBERT [LYY19]	76.66	70.47	77.08	80.05	71.05	75.60	49.46	58.52	62.78	66.72	74.09	99.22	50.62	99.07	81.78	70.94
MCAN [YYC19]	77.36	71.25	74.52	79.86	68.94	82.73	49.70	62.49	54.79	68.00	76.20	99.08	47.32	98.99	83.25	84.87
DFAF [GJY19]	77.72	72.17	78.28	81.80	70.68	81.69	51.42	**67.01**	56.60	67.72	77.60	99.02	50.27	98.83	84.11	85.70
UNITER [CLY20]	78.71	72.39	78.53	81.31	71.01	83.67	48.34	61.25	60.81	69.77	78.37	99.41	49.18	99.38	86.10	87.84
ViT [Won21]	79.15	72.34	78.92	82.60	70.84	82.12	54.64	68.80	58.46	68.66	77.41	98.95	51.10	98.76	84.72	86.07
ViLT [KSK21]	79.67	72.69	79.27	**82.61**	71.13	84.95	53.38	66.72	59.22	69.99	75.81	99.02	50.55	98.91	86.10	87.65
Patch-TRM (Ours)	**82.66**	**75.19**	**83.62**	81.87	**77.81**	**87.00**	**55.62**	62.39	**68.75**	**77.98**	**82.13**	98.24	**56.73**	97.98	**92.49**	**95.73**

Table 4.4: Results on the IconQA dataset.

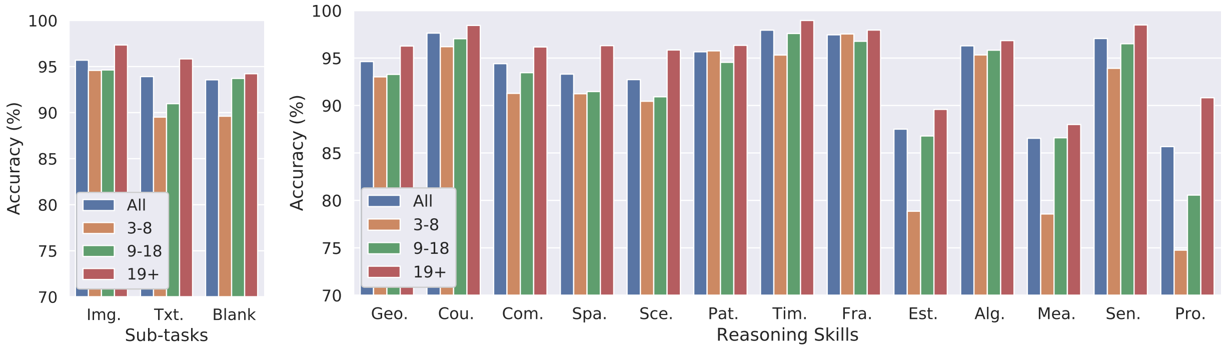


Figure 4.4: Performance of humans in different age groups for the IconQA task. Left: Accuracy over three sub-tasks; Right: Accuracy over thirteen reasoning skills.

are outperformed by models quite significantly in questions that require numerical reasoning skills like *probability*, *measurement*, and *estimation*.

	Sub-tasks (3)			**Reasoning skills (13)**												
Method	Img.	Txt.	Blank	Geo.	Cou.	Com.	Spa.	Sce.	Pat.	Tim.	Fra.	Est.	Alg.	Mea.	Sen.	Pro.
Human	95.69	93.91	93.56	94.63	97.63	94.41	93.31	92.73	95.66	97.94	97.45	87.51	96.29	86.55	97.06	85.67
Human (3-8)	94.58	89.51	89.61	93.02	96.20	91.28	91.24	90.45	95.76	95.32	97.54	78.86	95.33	78.57	93.92	74.76
Human (9-18)	94.63	90.97	93.71	93.28	97.04	93.46	91.47	90.92	94.55	97.59	96.77	86.79	95.83	86.60	96.51	80.56
Human (19+)	97.34	95.83	94.22	96.27	98.44	96.17	96.31	95.85	96.34	98.96	97.95	89.59	96.84	88.00	98.49	90.82

Table 4.5: Human performance in the IconQA task.

Analysis by task types. Humans outperform all benchmarks consistently over there sub-tasks and most reasoning skills. There is still a large gap to fill for future research of abstract diagram understanding and visual reasoning on the icon domain. The results achieved in blind studies of

Q-only and I-only are close to random, showing that the IconQA dataset is robust and reliable in distribution. Our proposed Patch-TRM baseline outperforms current state-of-the-art VQA models in all three sub-tasks. These improvements mainly come from two insights: pre-training ResNet on icon images and taking a hierarchical approach with attention mechanism.

Analysis by reasoning types. Similarly, the Patch-TRM baseline obtains better results than the benchmarks over most reasoning skill types. Interestingly, in some skills such as *estimation*, *measurement*, and *probability*, Patch-TRM performs better than average human beings. This implies that neural networks have a promising potential to develop the basic ability of mathematical reasoning.

4.5.4 Ablation Study

To study the functions of individual components in our model, we conduct an ablation analysis. Table 4.6 presents the results of different simplifications of our full model, where each implementation is trained on the IconQA train set and tested on the validation set. Instead of ResNet101 pre-trained on the icon classification task, *Patch-TRM w/o pre* utilizes ResNet101 pre-trained on natural image data for patch feature extraction. The decreasing performance of 0.95-2.49% indicates that pre-training backbones on tasks within similar domains is critical to downstream tasks. The attention mechanism helps to combine the image and question representations and improves the model performance by up to 7% compared to using simple concatenation (denoted as *Patch-TRM w/o att*). Positional embeddings of the ordered diagram patches benefit the vision Transformer by enabling it to learn spatial relationships among the patches, compared to the baseline without position embeddings (*Patch-TRM w/o pos*). *Patch-TRM V-CLS* uses the output embedding of `[CLS]` token as the diagram feature instead, which leads to a drastic performance decline. We have also experimented with coarse-grained patch cropping (e.g., *Pyramid 1+4+9+16* denotes 30 patches, *Pyramid 1+4+9* denotes 14 patches), which results in a performance degradation of 0.51% to 7.79%.

Method	Img.	Txt.	Blank
Patch-TRM w/o pre	82.01	72.72	81.67
Patch-TRM w/o att	80.63	68.00	80.29
Patch-TRM w/o pos	81.27	64.98	80.68
Patch-TRM V-CLS	80.15	63.90	70.27
Pyramid 1+4+9+16	82.45	68.76	82.19
Pyramid 1+4+9	80.61	67.42	81.36
Full model	**82.96**	**75.21**	**83.10**

Table 4.6: Ablation study in the IconQA dataset.

4.5.5 Quantitative Analysis

Quantitative analysis. We visualize one example with the cross-modal attention map generated by our baseline Patch-TRM in Figure 4.5. The visualized attention shows that our baseline is capable of attending to the corresponding patch regions with higher weights given the input question.

Q: Which object is next to the one shaped like a cube?

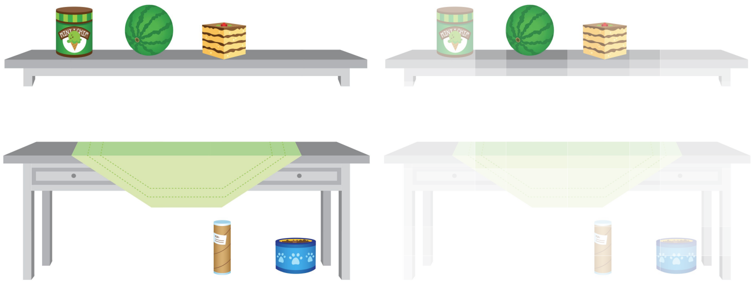


Figure 4.5: An example of text-to-image attention visualization by Patch-Trm.

Figure 4.6 presents five examples from the IconQA test set predicted by our Patch-TRM baseline for each sub-task. Although Patch-TRM achieves promising results for most problems in IconQA, it still fails to address some complicated cases. For example, it encounters difficulties in identifying dense objects and making multi-hop reasoning.

4.6 Conclusion

In this work, we introduce icon question answering (IconQA), a novel task that requires understanding abstract scenes and diverse visual reasoning skills. Unlike existing VQA methods relying

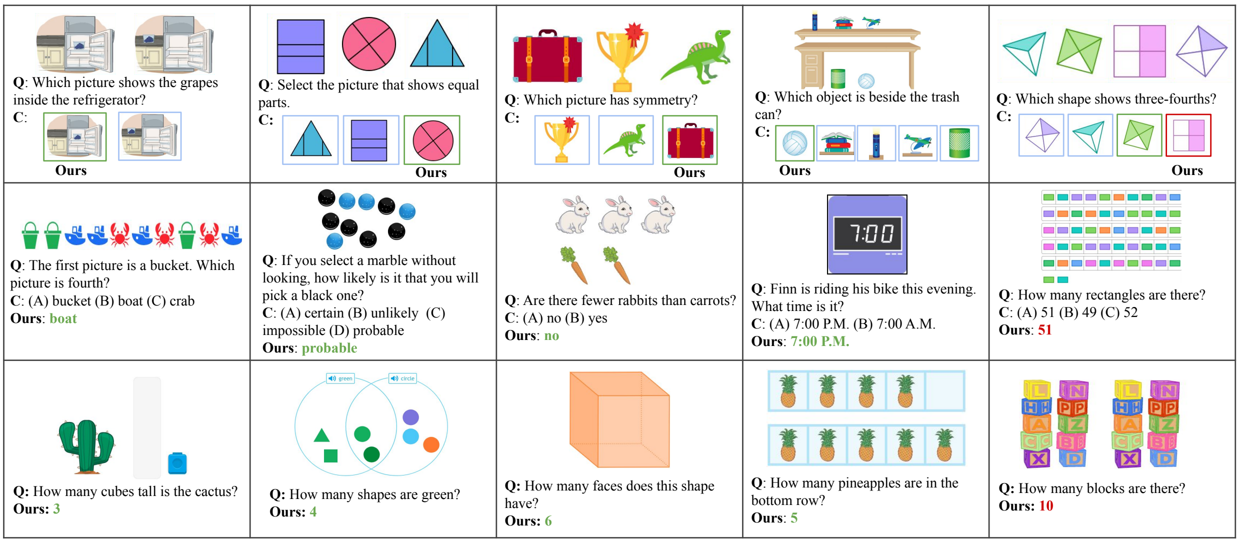

Q: Which picture shows the grapes inside the refrigerator?
C:
Ours
Q: Select the picture that shows equal parts.
C:
Ours
Q: Which picture has symmetry?
C:
Ours
Q: Which object is beside the trash can?
C:
Ours
Q: Which shape shows three-fourths?
C:
Ours
Q: The first picture is a bucket. Which picture is fourth?
C: (A) bucket (B) boat (C) crab
Ours: boat
Q: If you select a marble without looking, how likely is it that you will pick a black one?
C: (A) certain (B) unlikely (C) impossible (D) probable
Ours: probable
Q: Are there fewer rabbits than carrots?
C: (A) no (B) yes
Ours: no
7:00
Q: Finn is riding his bike this evening. What time is it?
C: (A) 7:00 P.M. (B) 7:00 A.M.
Ours: 7:00 P.M.
Q: How many rectangles are there?
C: (A) 51 (B) 49 (C) 52
Ours: 51
Q: How many cubes tall is the cactus?
Ours: 3
green
circle
Q: How many shapes are green?
Ours: 4
Q: How many faces does this shape have?
Ours: 6
Q: How many pineapples are in the bottom row?
Ours: 5
Q: How many blocks are there?
Ours: 10

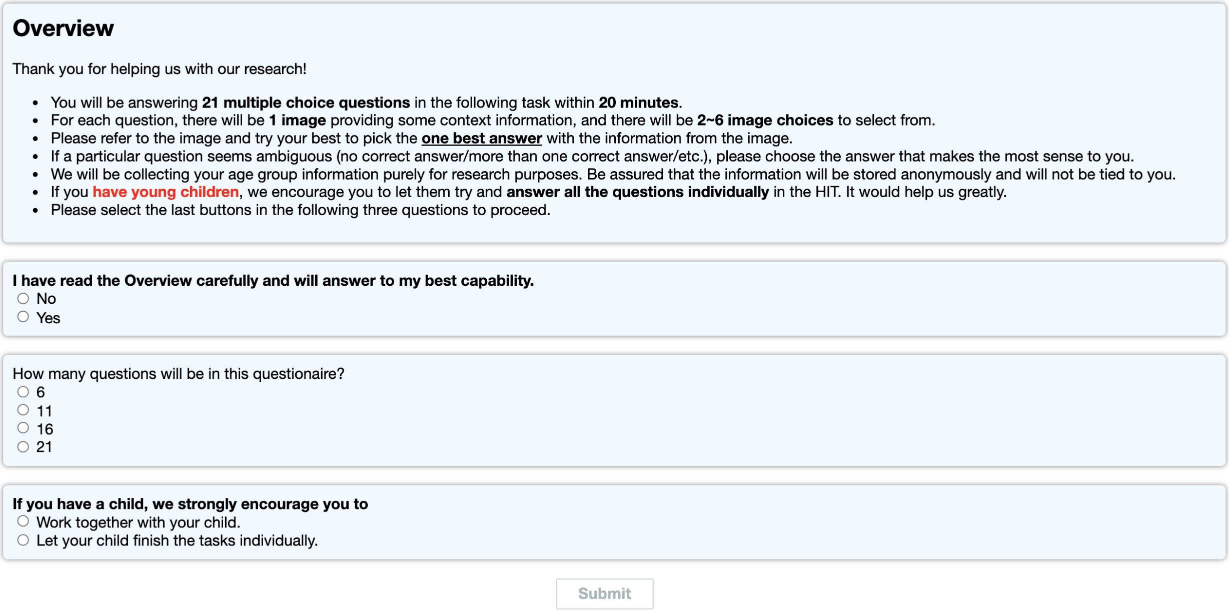
Figure 4.6: Result examples predicted by Patch-TRM in the IconQA test set. Top: *Multi-image-choice* sub-task. Middle: *Multi-text-choice* sub-task. Bottom: *Filling-in-the-blank* sub-task. Correctly predicted answers are highlighted in green, while wrong ones are highlighted in red.

on pre-trained models learned from natural images, we propose a new approach called Patch-TRM. It parses diagrams into patches using a pyramid layout and learns joint diagram-question representations via a cross-modal Transformer. We also pre-train the ResNet backbone on icon classification to enhance visual representations for this domain. Patch-TRM significantly outperforms previous VQA methods on IconQA. Future research could focus on improved diagram parsing and explicit visual reasoning for IconQA. Importantly, IconQA has promising potential applications in areas like online education, where visual reasoning over abstract diagrams is crucial.

4.7 Appendix: User Study

4.7.1 Crowd Sourcing Method

Using Amazon Mechanical Turk (AMT), we ask people to provide answers to the questions in the test set along with their age group. We also strongly encourage parents who have young children to let their children complete the questionnaires, as their answers give us insights to how the designed audience of these questions perform. The test set is split into batches of 20 questions, which we call a task, with each task assigned to 3 crowd workers on AMT. This amounts to a total of 64,467



Overview

Thank you for helping us with our research!

- You will be answering **21 multiple choice questions** in the following task within **20 minutes**.
- For each question, there will be **1 image** providing some context information, and there will be **2~6 image choices** to select from.
- Please refer to the image and try your best to pick the **one best answer** with the information from the image.
- If a particular question seems ambiguous (no correct answer/more than one correct answer/etc.), please choose the answer that makes the most sense to you.
- We will be collecting your age group information purely for research purposes. Be assured that the information will be stored anonymously and will not be tied to you.
- If you **have young children**, we encourage you to let them try and **answer all the questions individually** in the HIT. It would help us greatly.
- Please select the last buttons in the following three questions to proceed.

I have read the Overview carefully and will answer to my best capability.
○ No
○ Yes

How many questions will be in this questionaire?
○ 6
○ 11
○ 16
○ 21

If you have a child, we strongly encourage you to
○ Work together with your child.
○ Let your child finish the tasks individually.

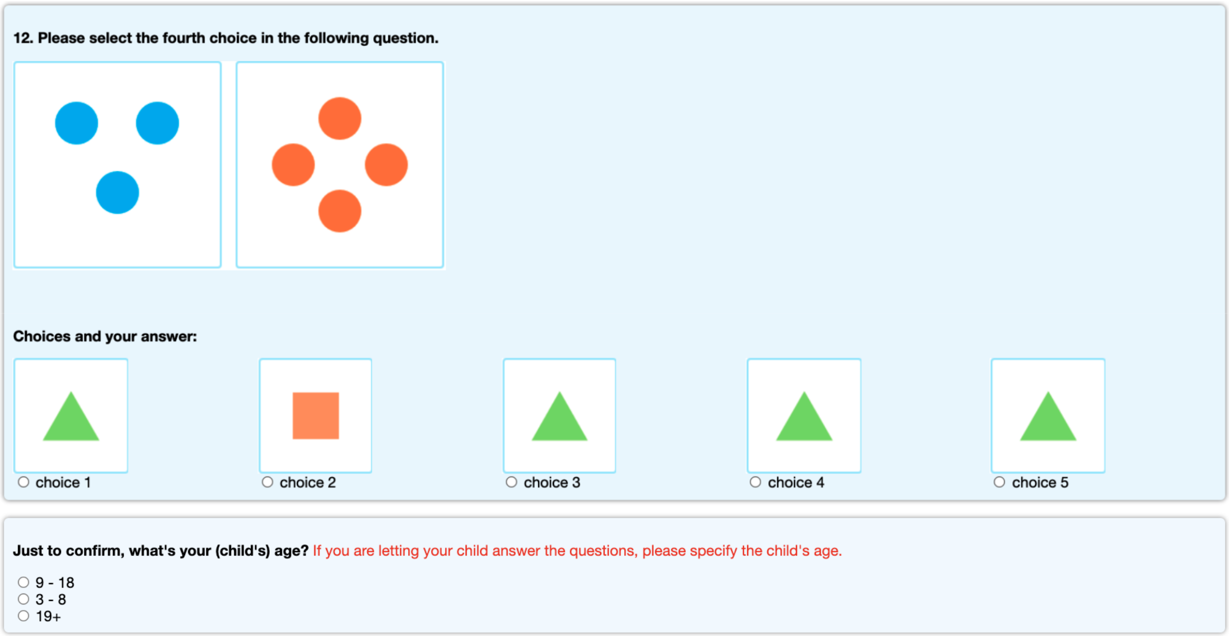
Figure 4.7: Instructions provided to AMT workers for the user study of the IconQA dataset.

effective test set answers.

4.7.2 Quality Assurance

To ensure the truthfulness of the age information, we ask the participants to select their age at both the beginning and the end of the questionnaire, with the age choices appearing in 2 different orders. To ensure the quality of the answers, we include 4 attention check questions: 3 of which are about the instructions, making sure that the participants read the instructions carefully. We also add an extra fake question in the middle for each *choosing an image choice* and *choosing a text choice* task, instructing them to choose the fourth choice despite what the choices are. Figure 4.7 shows the instructions and the first three attention check questions. Figure 4.8 shows the fake question along with the age confirmation. Figure 4.9 is an example question for the *multi-image-choice* sub-task. We also make sure that the workers answering our tasks have a history HIT approval rate of at least 95% and a previous approval count of 1,000.

In summary, for each Human Intelligence Task (HIT) on AMT, we have 2 age questions, 4 attention check questions, and 20 real questions from the IconQA test set. Among the 64,467 test answers, we filter out 1) the questionnaires that do not pass the 4 attention check questions, 2)



12. Please select the fourth choice in the following question.

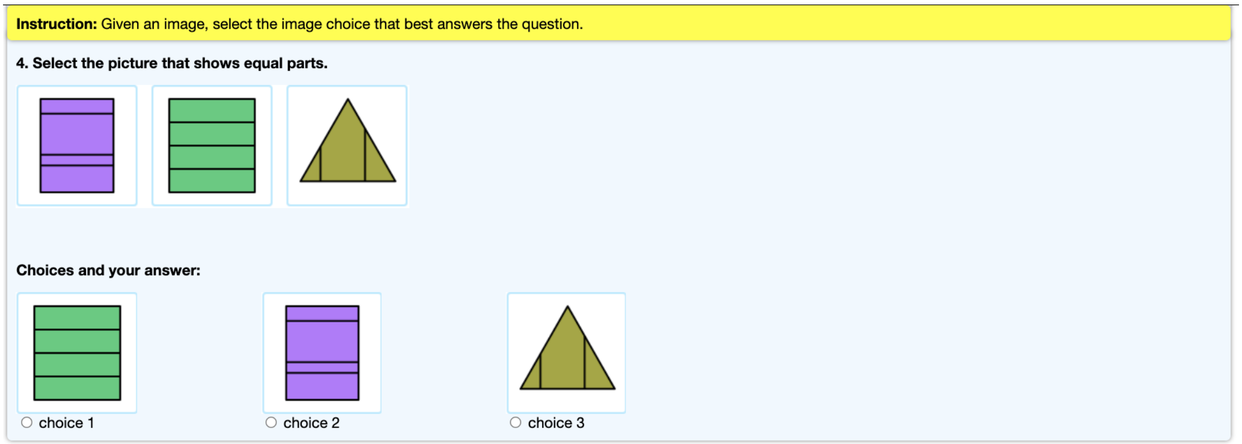
Choices and your answer:

○ choice 1 ○ choice 2 ○ choice 3 ○ choice 4 ○ choice 5

Just to confirm, what's your (child's) age? If you are letting your child answer the questions, please specify the child's age.

○ 9 - 18
○ 3 - 8
○ 19+

Figure 4.8: Attention check questions presented to AMT workers during the IconQA user study.



Instruction: Given an image, select the image choice that best answers the question.

4. Select the picture that shows equal parts.

Choices and your answer:

○ choice 1 ○ choice 2 ○ choice 3

Figure 4.9: An AMT example question from the *multi-image-choice* sub-task of IconQA.

the questionnaires that do not answer consistently for the two age-related questions, 3) the questionnaires that are finished unreasonably slowly/quickly. After filtering, we have 54,896 effective question answers, which we believe is a decently large sample for the human performance study.

Part III

Retrieval and Tool-Augmented Algorithms

CHAPTER 5

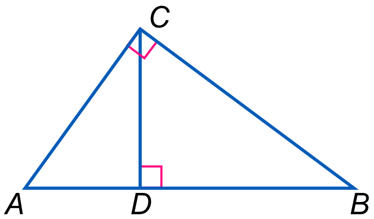
Neuro-Symbolic Geometric Solver

5.1 Introduction

Geometry problem solving is a long-standing challenging task in artificial intelligence and has been gaining more attention in the NLP community recently [SHF14, HLP19, SDH20]. Solving geometry problems is an essential subject in high-school education for the development of students' abstract thinking. As an example shown in Figure 5.1, given problem text in natural language and a corresponding diagram, one needs to identify the geometric relations, apply theorem knowledge, and conduct algebraic calculations to derive the numerical value of the answer.

Psychologists and educators believe that solving geometric problems requires high-level thinking abilities of symbolic abstraction and logical reasoning [Chi98, NN17]. However, if algorithms take the raw problem content, it might encounter challenges to understand the abstract semantics and perform human-like cognitive reasoning for inferring the answer in the geometry domain. A formal language is composed of words from a well-formed alphabet based on a specific set of rules and is commonly used in the fields of linguistics and mathematics. Therefore, our proposed geometry solver parses the problem inputs into formal language descriptions (see examples in Figure 5.1) before solving the problems.

To translate the problem text and diagrams to formal descriptions, existing methods [SHF15, SDX17, SX17] highly depend on human annotations like symbols in diagrams as the intermediate results. Also, these methods fail to provide the explicit reasoning processes when predicting the answer. For example, [SHF15] simplifies the problem solving task to an optimization problem to pick one that satisfies all constraints from choice candidates. Furthermore, most current datasets are either small in scale or not publicly available [SHF15, SX17], which further hinders



In triangle *ABC*, *AD* = 3 and *BD* = 14. Find *CD*.
Choices:
A. 6.0 B. 6.5 C. 7.0 D. 8.5
Answer: B

Diagram Formal Language

```
Triangle(A,B,C)
Triangle(A,C,D)
Triangle(B,C,D)
PointLiesOnLine(D,Line(A,B))
Perpendicular(Line(A,C),Line(B,C))
Perpendicular(Line(C,D),Line(A,B))
```

Text Formal Language

```
Triangle(A,B,C)
Equals(LengthOf(Line(A,D)),3)
Equals(LengthOf(Line(B,D)),14)
Find(LengthOf(Line(C,D)))
```

Figure 5.1: A data example from the Geometry3K dataset, annotated with formal language descriptions for both the problem text and diagram.

the research of geometry problem solving.

To overcome these challenges, we have constructed a new large-scale benchmark, called Geometry3K, to assess algorithms' performance of geometry problem solving. The Geometry3K dataset consists of 3,002 multi-choice problems as well as covers diverse geometric shapes and problem goals (See details in 2.1). Each problem in Geometry3K is annotated with the problem text, a diagram, four choices, and a correct answer. In contrast with existing work, we also annotate each problem text and diagram with unified structural descriptions in formal language.

We further present a novel geometry solving approach with formal language and symbolic reasoning, called *Interpretable Geometry Problem Solver* (Inter-GPS). Inter-GPS (Figure 5.2) develops an automatic parser that translates the problem text via template rules and parses diagrams by a neural object detector into formal language, respectively. In contrast to parameter learning, Inter-GPS formulates the geometry solving task as problem goal searching, and incorporates theorem knowledge as conditional rules to perform symbolic reasoning step by step. It demonstrates an interpretable way to tackle the task. Also, we design a theorem predictor to infer the possible theorem application sequence in Inter-GPS for the efficient and reasonable searching path. Extensive experiments on the Geometry3K and GEOS datasets show Inter-GPS achieves large improvements over existing methods.

Our contributions are three-fold: (1) we introduce a large-scale diverse benchmark of geometry problem solving, Geometry3K, which is densely annotated with formal language; (2) we develop an automatic problem parser to translate the problem text and diagram into formal language; (3) we propose a novel interpretable problem solver that applies symbolic reasoning to infer the answer.

5.2 Related Work

Approaches for Geometry Problem Solving. Due to the sparsity of appropriate data, most early works on automated geometry systems focus on geometry theorem proving [Wen86, CGZ96, YWG19, GYZ19], problem synthesis [AGM14], diagram parsing [SHF14], as well as problem formalization [GY18]. [SHF15] attempt using computer vision and natural language processing techniques to solve geometry problems with problem understanding. However, the system does not perform explicit reasoning with axiomatic knowledge as it reduces the task to an optimization problem to see which choice can satisfy all constraints. Some recent efforts [SDX17, SDH20] have been made to incorporate theorem knowledge into problem solving. They feed geometry axioms written as horn clause rules and declarations from the diagram and text parser into logical programs in prolog style to solve the problem. However, these methods fail to provide human-readable solving steps. And parameter learning on horn clause rules and built-in solvers leads to an uncontrollable search process. In contrast, our proposed Inter-GPS implements explicit symbolic reasoning to infer the answer without the help of candidate answers in an interpretable way.

Interpretable Math Problem Solving. Due to the intrinsic requirements of symbolic understanding and logical reasoning, interpretability of solvers plays an essential role in geometry problem solving. While the interpretability of geometry problem solvers is rarely explored, some pioneering work has been proposed in the general math problem solving domain. Broadly there are two main lines of achieving interpretable solving steps for math problems. The first generates intermediate structural results of equation templates [HSL17, WZZ19], operational programs [AGL19] and expression trees [WWC18, QLL20, HLC21]. The second line of work with a higher level of interpretability translates the math problems into symbolic language and conducts logical reasoning iteratively to predict the final results [MII17, RR18]. Furthermore, inspired by work on semantic parsing [HZ05, ZM06, TML14], we claim structured diagram parsing and joint semantic representations for text and diagrams is critical in interpretable geometry problem solving.

5.3 Geometry Formal Language

A geometry problem P is defined as a tuple $(t, d, \mathbf{c})$, in which t is the input text, d is the diagram image and $\mathbf{c} = \{c_1, c_2, c_3, c_4\}$ is the multiple-choice candidate set in the format of numerical values. Given the text t and diagram d, an algorithm is required to predict the correct answer $c_i \in \mathbf{c}$. We formally describe the problem in the geometric domain language Ω, a set of literals composed of predicates and arguments. Basic terms used in the geometry problem solver are defined as follows.

Definition 1. A *predicate* is a geometric shape entity, geometric relation, or arithmetic function.

Definition 2. A *literal* is an application of one *predicate* to a set of arguments like variables or constants. A set of *literals* makes up the semantic description from the problem text and diagrams in the formal language space Ω.

Definition 3. A *primitive* is a basic geometric element like a point, a line segment, a circle, or an arc segment extracted from the diagram.

Table 5.1 lists examples of *predicates* and *literal* templates. There are 91 *predicates* in our defined formal language, and we list them in the Tables 5.7 to 5.12 in Appendix 5.8.

Terms	**Examples**
predicate	`Line`, `IntersectAt`, `IsMedianOf`
literal	`Find(AreaOf(Triangle(A,B,C))`

Table 5.1: Examples of terms used in the formal language for geometry problem solving.

5.4 Geometry Problem Parser

Our proposed Inter-GPS takes the problem text and diagrams as inputs and translates them into formal language descriptions automatically via the text parser (Section 5.4.1) and the diagram parser (Section 5.4.2), respectively.

5.4.1 Text Parser

Given the word sequence of the problem text T, the text parser needs to translate it into a set of literals L_t, a sequence composed of predicates and variables. Recently, deep neural networks have achieved promising performances in sequence-to-sequence (Seq2Seq) learning tasks like machine translation [SVL14, VSP17, KT19]. However, semantic parsers using Seq2Seq learning methods are not feasible to generate satisfactory literals in the Geometry3K dataset for two reasons. Firstly, the limited scale of geometry datasets weakens these highly data-driven methods. Secondly, neural semantic parsers tend to bring noises in generated results while geometry solvers with symbolic reasoning are sensitive to such deviations.

Inspired by previous works [KCC08, SHF15, BGL14] that indicate the rule-based parsing method is able to obtain precise parsing results, we apply this approach with regular expressions to perform text parsing. We also achieve a semantic text parser using BART [LLG20], one of the state-of-the-art sequence learning models for comparison.

5.4.2 Diagram Parser

Diagrams provide complementary geometric information that is not mentioned in the problem text. Previous works [SHF14, SHF15] require manual annotations to identify symbols in the diagrams and fail to deal with special relational symbols such as *parallel*, *perpendicular*, and *isosceles*. Instead, an automatic diagram parser without human intervention is proposed in this section and is able to detect varied diagram symbols.

The diagram parser first applies Hough Transformation [SS01] to extract geometry primitives (points, lines, arcs, and circles), following [SHF15]. Then the diagram symbols and text regions are extracted through a strong object detector RetinaNet [LGG17], and the textual content is further recognized by the optical character recognition tool MathPix[1]. After obtaining the primitive set P and symbol set S, we need to ground each symbol with its associated primitives. [SHF15] adapts a greedy approach where each symbol is assigned to the closest primitive without considering its

---

[1] `https://mathpix.com/`

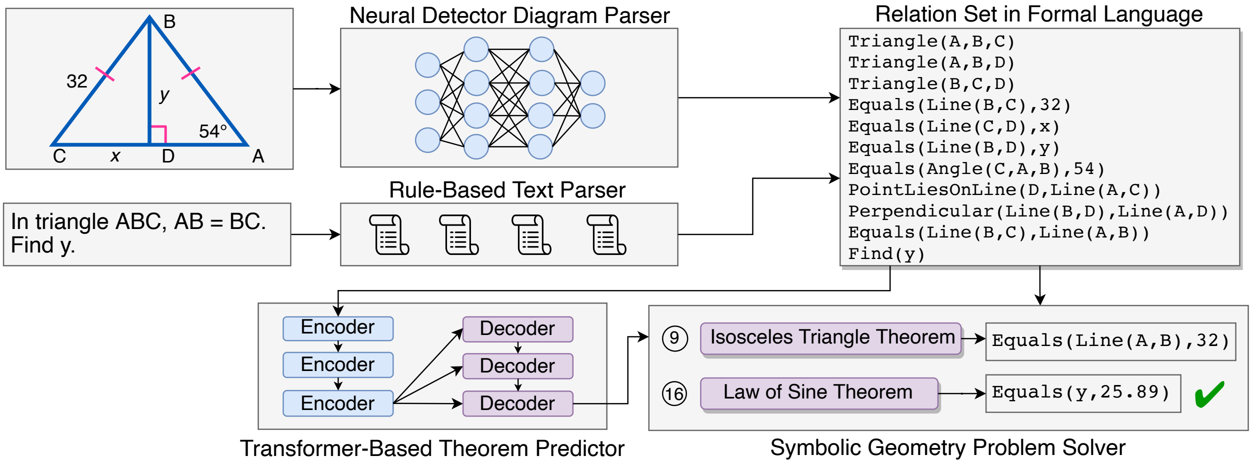
validity. Instead, we formulate the grounding task as an optimization problem with the constraint of geometry relations:

$$\begin{aligned} &\min \sum_s dist(s_i, p_j) \times \mathbb{1}\{s_i \text{ assigns to } p_j\} \\ &\text{s.t. } (s_i, p_j) \in \text{Feasibility set } F, \end{aligned} \tag{5.1}$$

where the $dist$ function measures the Euclidean distance between the symbol s_i and primitive p_j. F defines the geometric constraints for symbol grounding. For example, the *parallel* symbol could only be assigned to two lines with the same slopes and the *perpendicular* symbol is only valid to two orthogonal lines.

5.5 Geometry Problem Solver

Unlike existing methods [SHF15, SDX17, AGM17, SDH20], Inter-GPS achieves the explicit symbolic reasoning with the theorem knowledge base and the human-readable search process, shown in Figure 5.2.


B
32
y
54°
C
x
D
A
In triangle ABC, AB = BC.
Find y.
Neural Detector Diagram Parser
Rule-Based Text Parser
Relation Set in Formal Language
Encoder
Encoder
Encoder
Decoder
Decoder
Decoder
Transformer-Based Theorem Predictor
⑨
Isosceles Triangle Theorem
⑯
Law of Sine Theorem
✓
Symbolic Geometry Problem Solver

```
Triangle(A,B,C)
Triangle(A,B,D)
Triangle(B,C,D)
Equals(Line(B,C),32)
Equals(Line(C,D),x)
Equals(Line(B,D),y)
Equals(Angle(C,A,B),54)
PointLiesOnLine(D,Line(A,C))
Perpendicular(Line(B,D),Line(A,D))
Equals(Line(B,C),Line(A,B))
Find(y)
```

```
Equals(Line(A,B),32)
Equals(y,25.89)
```

Figure 5.2: Overview of the Inter-GPS architecture. Given the problem diagram and text, Inter-GPS first parses the inputs into a relation set defined in formal language. Then it applies the theorem sequence predicted by the theorem predictor to perform symbolic reasoning over the relation set to infer the answer. ⑨ and ⑯ denote the orders of the applied theorems.

5.5.1 Symbolic Geometry Solver

Overall, Inter-GPS takes the relation set $\mathcal{R}$ and the theorem knowledge base set $\mathcal{KB}$ as inputs, and outputs the numeric solution g^* of the problem goal g. The relation set $\mathcal{R}$ defines geometry attributes and relations in the given problem, and is initialized with literals from the text and diagram parsers. $\mathcal{R}$ is further expanded with literals that are derived from definitions of geometry shapes. For example, a triangle is defined as three connected sides. So if there is a *literal* `Triangle(A,B,C)`, six more *literals* (`Ponit(A)`, `Ponit(B)`, `Ponit(C)`, `Line(A,B)`, `Line(B,C)`, `Line(C,A)`) will be appended to $\mathcal{R}$.

The theorem set $\mathcal{KB}$ is represented as a set of theorems, where each theorem k_i is written as a conditional rule with a premise p and a conclusion q. For the search step t, if the premise p of k_i matches the current relation set $\mathcal{R}_{t-1}$, the relation set is updated according to the conclusion q:

$$\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}, k_i \in \mathcal{KB}. \tag{5.2}$$

After the application of several theorems, equations between the known values and the unknown problem goal g are established, and g could be solved after solving these equations:

$$g^* \leftarrow \textsc{SolveEquation}(\mathcal{R}_t, g). \tag{5.3}$$

5.5.2 Theorem Predictor (TP)

As the geometry problems in Geometry3K are collected from high school textbooks, it might need to apply multiple theorems before the problems are solved. Intuitively, one possible search strategy is to use brute force to enumerate candidates in the theorem set randomly. The random search strategy is inefficient and might lead to problems unsolvable as there might be applications of complicated theorems in the early stage. Therefore, an ideal geometry problem solver can solve the problems using reasonable theorem application sequences. Students with good academic performance can solve a problem with prior knowledge learning from a certain amount of problem solving training. Inspired by this phenomenon, a theorem predictor is proposed to infer the possible theorem application sequence for inference after multiple attempts on the train data. Recent studies [LIS17, BBV18] also suggest that neural guided search can speed up the search process.

There are no annotated theorem application sequences for data in Geometry3K due to tremendous worker labor. Thus, we randomly sample from the theorem set multiple times to generate the application sequences. A generated sequence is regarded as positive if the geometry solver Inter-GPS solves the problem after the application of that sequence. A positive sequence with the minimum length for a problem is seen as pseudo-optimal. Finally, after attempts, we collect 1,501 training samples with the problem and its pseudo-optimal theorem application sequence.

Given the problem formal description $L = \{l_1, ..., l_m\}$, the theorem predictor aims to reconstruct the pseudo-optimal theorem sequence $T = \{t_1, ..., t_n\}$ token by token. We formulate the generation task as a sequence-to-sequence (Seq2Seq) problem and use a transformer-based model [LLG20] to generate theorem sequence tokens. Specifically, the transformer decoder predicts the next theorem order t_i given $T = \{t_1, ..., t_i\}$. The Seq2Seq model is trained to optimize the negative log-likelihood loss:

$$\mathcal{L}_{\mathrm{TP}} = -\sum_{i=1}^{n} \log p_{\mathrm{TP}} \left(t_i \mid t_1, \ldots, t_{i-1}\right), \tag{5.4}$$

where p_{TP} is the parametrized conditional distribution in the theorem predictor model.

5.5.3 Low-first Search Strategy

After the application of the theorem sequence predicted by the theorem predictor, it is likely that Inter-GPS still could not find the problem goal. Generally, humans incline to use simple theorems first when solving math problems to reduce complex calculations. If simple theorems are not tangible, they will turn to more complex theorems. On account of that, we apply an efficient search strategy with heuristics driven by subject knowledge. We categorize theorems into two groups: **lower-order** theorem set $\mathcal{KB}_1$ and **higher-order** theorem set $\mathcal{KB}_2$. The lower-order set $\mathcal{KB}_1$ (e.g, *Triangle Angle-Sum Theorem*, *Congruent Triangle Theorem*) only involves in two simple operations of addition and subtraction, while $\mathcal{KB}_2$ (e.g, *Law of Sines*) requires complex calculations.

Algorithm 1 Symbolic Geometry Solver

```
Input: Literals $\mathcal{L}$, goal $g$, knowledge bases $\mathcal{KB}_1$, $\mathcal{KB}_2$
Onput: Numeric goal value $g^*$ and theorem application $\mathcal{S}$
function SEARCH($\mathcal{L}$, $g$, $\mathcal{KB}_1$, $\mathcal{KB}_2$)
    Initialize relation set $\mathcal{R}_0$ with $\mathcal{L}$, $g^* = \emptyset$, $\mathcal{S} = \emptyset$
    $\mathcal{KB}_p \leftarrow$ THEOPREDICTOR($\mathcal{L}$)    ▷ Predicted
    for $k_i \in \mathcal{KB}_p$ do
        $\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}$
        $\mathcal{S}$.APPEND($k_i$)
    end for
    $g^* \leftarrow$ SOLVEEQUATION($\mathcal{R}_t, g$)
    if $g^* \neq \emptyset$ then
        return $g^*$ and $S$
    end if
    while $g^* = \emptyset$ and $\mathcal{R}_t$ is updated do
        for $k_i \in \mathcal{KB}_1$ do    ▷ Lower-order
            $\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}$
            $\mathcal{S}$.APPEND($k_i$)
            $g^* \leftarrow$ SOLVEEQUATION($\mathcal{R}_t, g$)
            if $g^* \neq \emptyset$ then
                return $g^*$ and $S$
            end if
        end for
        for $k_i \in \mathcal{KB}_2$ do    ▷ Higher-order
            $\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}$
            $\mathcal{S}$.APPEND($k_i$)
            $g^* \leftarrow$ SOLVEEQUATION($\mathcal{R}_t, g$)
            if $g^* \neq \emptyset$ then
                return $g^*$ and $S$
            end if
        end for
    end while
end function
```

In each following search step after using predicted theorems, we first enumerate theorems in the lower-order set $\mathcal{KB}_1$ to update the relation set $\mathcal{R}$:

$$\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}, k_i \in \mathcal{KB}_1. \tag{5.5}$$

If lower-order theorems fail to update $\mathcal{R}$ anymore, higher-order theorems are considered to update $\mathcal{R}$:

$$\mathcal{R}_t \leftarrow k_i \wedge \mathcal{R}_{t-1}, k_i \in \mathcal{KB}_2. \tag{5.6}$$

The search process stops once we find the problem goal g or the search steps reach the maximum steps allowed. The whole search algorithm for Inter-GPS is presented in Algorithm 1.

5.6 Experiments

5.6.1 Experimental Settings

Datasets and evaluation metrics. We conduct experiments on the Geometry3K and GEOS [SHF15] datasets. The Geometry3K dataset involves 2,101 training data, 300 validation data, and 601 test data, respectively. The GEOS dataset provides 55 official SAT problems for evaluating geometry solvers. Regarding our proposed Inter-GPS model, if the one closest to the found solution among the four choices is exactly the ground truth, the found solution is considered correct. For a fair comparison, if Inter-GPS fails to output the numeric value of the problem goal within allowed steps, it will randomly choose the one from the four candidates. In terms of compared neural network baselines, the predicted answer has a maximum confidence score among choice candidates.

Baselines. We implement several deep neural network baselines for geometry solvers to compare them with our method. By default, these baselines formalize the geometry problem solving task as a classification problem, fed by the text embedding from a sequence encoder and the diagram representation from a visual encoder. *Q-only* only encodes the problem text in the natural language by a bi-directional Gated Recurrent Unit (Bi-GRU) encoder [CMB14]. *I-only* only encodes the problem diagram by a ResNet-50 encoder [HZR16] as the input. *Q+I* uses Bi-GRU and ResNet-50 to encode the text and diagram, respectively. *RelNet* [BNM17] is implemented for embedding the problem text because it is a strong method for modeling entities and relations. *FiLM* [PSD18] is compared as it achieves effective visual reasoning for answering questions about abstract images.

Method	**All**	**Angle**	**Length**	**Area**	**Ratio**	**Line**	**Triangle**	**Quad**	**Circle**	**Other**
Random	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0	25.0
Human	56.9	53.7	59.3	57.7	42.9	46.7	53.8	68.7	61.7	58.3
Human Expert	90.9	89.9	92.0	93.9	66.7	95.9	92.2	90.5	89.9	92.3
Q-only	25.3	29.5	21.5	28.3	33.3	21.0	26.0	25.9	25.2	22.2
I-only	27.0	26.2	28.4	24.5	16.7	24.7	26.7	30.1	30.1	25.9
Q+I	26.7	26.2	26.7	28.3	25.0	21.0	28.1	32.2	21.0	25.9
RelNet [BNM17]	29.6	26.2	34.0	20.8	41.7	29.6	33.7	25.2	28.0	25.9
FiLM [PSD18]	31.7	28.7	32.7	**39.6**	33.3	33.3	29.2	33.6	30.8	29.6
FiLM-BERT [KT19]	32.8	32.9	33.3	30.2	25.0	32.1	32.3	32.2	34.3	33.3
FiLM-BART [LLG20]	33.0	32.1	33.0	35.8	50.0	34.6	32.6	37.1	30.1	37.0
Inter-GPS (ours)	**57.5**	**59.1**	**61.7**	30.2	**50.0**	**59.3**	**66.0**	**52.4**	**45.5**	**48.1**
Inter-GPS (GT)	78.3	83.1	77.9	62.3	75.0	86.4	83.3	77.6	61.5	70.4

Table 5.2: Evaluation results of our proposed method and comparison with baselines on the Geometry3K dataset.

FiLM-BERT uses the BERT encoder [KT19] instead of the GRU encoder, and *FiLM-BART* uses the recently proposed BART encoder [LLG20].

Implementation details. Main hyper-parameters used in the experiments are shown below. For our symbolic solver, a set of 17 geometry theorems is collected to form the knowledge base. For generating positive theorem sequences, each problem is attempted by 100 times with the maximum sequence length of 20. The transformer model used in the theorem predictor has 6 layers, 12 attention heads, and a hidden embedding size of 768. Search steps in Inter-GPS are set up to 100. For the neural solvers, we choose the Adam optimizer and set the learning rate as 0.01, and the maximum epochs are set as 30. Each experiment for Inter-GPS is repeated three times for more precise results.

5.6.2 Comparisons with Baselines

Table 5.2 compares the results of symbolic solver Inter-GPS with baselines on our proposed Geometry3K dataset. Apart from the overall accuracy, the results of different problem types are also reported. Benefiting from symbolic reasoning with theorem knowledge, our Inter-GPS obtains an overall accuracy of 57.5%, significantly superior to all neural baselines. Inter-GPS even attains a better accuracy compared to human beings. Inter-GPS with ground truth formal language gains a further improvement of 20.8%. Inter-GPS also obtains state-of-the-art performance over exiting

Method	**Acc (%)**
GEOS [SHF15]	49
GEOS++ [SHF15]	49
GEOS-OS [SX17]	52
GEOS++AXIO [SDX17]	55
Inter-GPS (ours)	**67**

Table 5.3: Evaluation results on the GEOS dataset.

geometry solvers on the GEOS dataset, as shown in Table 5.3.

5.6.3 Ablation Study and Discussion

Search strategies. The overall accuracy and average steps needed for solving problems with different search strategies in Inter-GPS are reported in Table 5.4. *Predict* refers to the strategy that uses the theorems from the theorem predictor followed by a random theorem sequence. The strategy largely reduces the average steps to 6.5. The final strategy in Inter-GPS applies the predicted theorems first and lower-order theorems in the remain search steps, and gains the best overall accuracy.

Search strategies	**Accuracy (%)**	**# Steps**
Random	75.5 ± 0.2	13.2 ± 0.1
Low-first	77.3 ± 0.3	15.1 ± 0.2
Predict	77.5 ± 0.1	**6.5** ± 0.1
Predict+Low-first (final)	**78.3** ± 0.1	7.1 ± 0.1

Table 5.4: Performance of Inter-GPS with different search strategies.

Problem parsers and literal sources. The rule-based text parser achieves an accuracy of 97% while only 67% for the semantic text parser. Table 5.5 reports the Inter-GPS performance fed with different sources of literals. With literals generated from our problem solver, Inter-GPS achieves an accuracy of 57.5%. The current text parser performs very well as there is only a slight gap between Inter-GPS with generated text literals and ground truth literals. An improvement of 17.5% for Inter-GPS with annotated diagram literals indicates that there is still much space to improve for the diagram parser.

Searching step distribution. Figure 5.3 compares correctly solved problem distribution by the av-

	Diagram w/o	**Diagram**	**Diagram (GT)**
Text w/o	25.0 ± 0.0	46.6 ± 0.7	58.7 ± 0.2
Text	25.4 ± 0.0	57.5 ± 0.2	75.0 ± 0.6
Text (GT)	25.4 ± 0.0	58.0 ± 1.7	**78.3** ± 0.1

Table 5.5: Performance of Inter-GPS with predicted and ground truth (GT) literals.

erage number of search steps in different strategies. Our final Inter-GPS applies the *Predict+Low-first* strategy, with which 65.97% problems are solved in two steps and 70.06% solved in five steps.

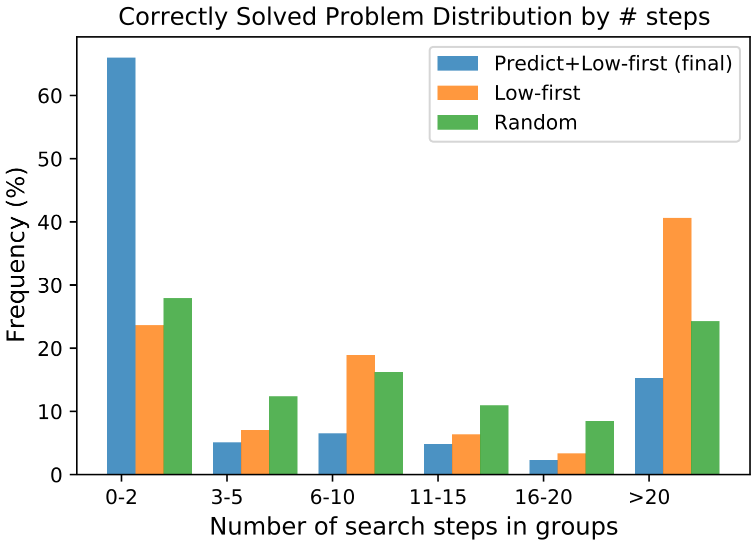


Figure 5.3: Distribution of correctly solved problems by the number of search steps required by Inter-GPS.

Neural geometry solvers. Current neural network baselines for geometry solving fail to achieve satisfactory results in the Geometry3K dataset. It is because there are limited data samples for these neural methods to learn meaningful semantics from the problem inputs. Besides, dense implicit representations might not be suitable for logical reasoning tasks like geometry problem solving. We replace the inputs of problem text and diagram in the *Q+I* baseline with the ground truth textual and visual formal annotations and report the result in Table 5.6. An improvement of 9.2% indicates the promising potential for neural network models for problem solving if structural representations with rich semantics are learned.

Failure cases. Inter-GPS might not find a solution because of inaccurate parsing results and the incomplete theorem set. Figure 5.4 illustrates some failure examples for Inter-GPS. For example,

	Diagram (visual)	**Diagram (formal)**
Text (natural)	26.7	35.3
Text (formal)	34.6	35.9

Table 5.6: Neural solver performance with different representations of the problem text and diagrams.

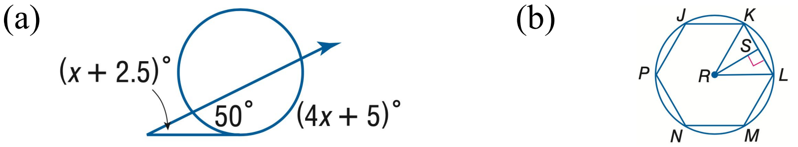
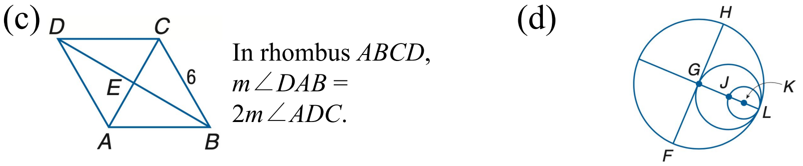


Diagram parser: ambiguous symbols Diagram parser: multiple primitives



Text parser: nested expressions Symbolic solver: complex theorems

Figure 5.4: Examples of failure cases encountered by Inter-GPS on the Geometry3K dataset.

diagram parsing tends to fail if there are ambiguous annotations or multiple primitives in the diagram. It is difficult for the text parser to handle nested expressions and uncertain references. And the symbolic solver is still not capable of solving complex problems with combined shapes and shaded areas in the diagrams.

Interpretability in Inter-GPS. Inter-GPS provides an interpretable symbolic solver for geometry problem solving. First, Inter-GPS parses the problem contents into a structural representation of formal language. Second, Inter-GPS performs symbolic reasoning to update the geometric relation set explicitly. Last, Inter-GPS applies reasonable theorems sequentially in the search process.

5.7 Conclusion

Solving geometry problems is one of the most challenging tasks in math question answering. We propose a large-scale benchmark, Geometry3K, which consists of 3,002 high-school geometry problems with dense descriptions in formal language. We further propose a novel geometry solving approach, *Interpretable Geometry Problem Solver* (Inter-GPS), which parses the problem as formal language from an automatic parser and performs symbolic reasoning over the theorem knowledge base to infer the answer. Also, a theorem predictor with a low-first search strategy is designed to generate the reasonable theorem application sequence. Experiment results show that Inter-GPS outperforms existing state-of-the-art methods by a large margin.

5.8 Appendix: Gemetric Formal Language

We define 91 *predicates* and their corresponding *literal* templates in the geometry language domain. For development, these *predicates* are categorized into six groups: geometric shapes (Table 5.7), unary geometric attributes (Table 5.8), general geometric attributes (Table 5.9), binary geometric relations (Table 5.10), A-IsXOf-B-type geometric relations (Table 5.11), as well as numerical attributes and relations (Table 5.12). Moreover, `$` in the literal templates denotes the undetermined shape.

#	**Predicates**	**Literal templates**
1	`Point`	`Point(A),Point($)`
2	`Line`	`Line(A,B),Line(m),Line($)`
3	`Angle`	`Angle(A,B,C),Angle(A),Angle(1),Angle($)`
4	`Triangle`	`Triangle(A,B,C),Triangle($),Triangle($1,$2,$3)`
5	`Quadrilateral`	`Quadrilateral(A,B,C,D),Quadrilateral(1),Quadrilateral($)`
6	`Parallelogram`	`Parallelogram(A,B,C,D),Parallelogram(1),Parallelogram($)`
7	`Square`	`Square(A,B,C,D),Square(1),Square($)`
8	`Rectangle`	`Rectangle(A,B,C,D),Rectangle(1),Rectangle($)`
9	`Rhombus`	`Rhombus(A,B,C,D),Rhombus(1),Rhombus($)`
10	`Trapezoid`	`Trapezoid(A,B,C,D),Trapezoid(1),Trapezoid($)`
11	`Kite`	`Kite(A,B,C,D),Kite(1),Kite($)`
12	`Polygon`	`Polygon($)`
13	`Pentagon`	`Pentagon(A,B,C,D,E),Pentagon($)`
14	`Hexagon`	`Hexagon(A,B,C,D,E,F),Hexagon($)`
15	`Heptagon`	`Heptagon(A,B,C,D,E,F,G),Heptagon($)`
16	`Octagon`	`Octagon(A,B,C,D,E,F,G,H),Octagon($)`
17	`Circle`	`Circle(A),Circle(1),Circle($)`
18	`Arc`	`Arc(A,B),Arc(A,B,C),Arc($)`
19	`Sector`	`Sector(O,A,B),Sector($)`
20	`Shape`	`Shape($)`

Table 5.7: 20 predicates and corresponding literal templates for geometric shapes.

#	**Predicates**	**Literal templates**
1	`RightAngle`	`RightAngle(Angle($))`
2	`Right`	`Right(Triangle($))`
3	`Isosceles`	`Isosceles(Polygon($))`
4	`Equilateral`	`Equilateral(Polygon($))`
5	`Regular`	`Regular(Polygon($))`
6	`Red`	`Red(Shape($))`
7	`Blue`	`Blue(Shape($))`
8	`Green`	`Green(Shape($))`
9	`Shaded`	`Shaded(Shape($))`

Table 5.8: 9 predicates and corresponding literal templates for unary geometric attributes.

#	**Predicates**	**Literal templates**
1	`AreaOf`	`AreaOf(A)`
2	`PerimeterOf`	`PerimeterOf(A)`
3	`RadiusOf`	`RadiusOf(A)`
4	`DiameterOf`	`DiameterOf(A)`
5	`CircumferenceOf`	`CircumferenceOf(A)`
6	`AltitudeOf`	`AltitudeOf(A)`
7	`HypotenuseOf`	`HypotenuseOf(A)`
8	`SideOf`	`SideOf(A)`
9	`WidthOf`	`WidthOf(A)`
10	`HeightOf`	`HeightOf(A)`
11	`LegOf`	`LegOf(A)`
12	`BaseOf`	`BaseOf(A)`
13	`MedianOf`	`MedianOf(A)`
14	`IntersectionOf`	`IntersectionOf(A,B)`
15	`MeasureOf`	`MeasureOf(A)`
16	`LengthOf`	`LengthOf(A)`
17	`ScaleFactorOf`	`ScaleFactorOf(A,B)`

Table 5.9: 17 predicates and corresponding literal templates for general geometric attributes .

#	**Predicates**	**Literal templates**
1	`PointLiesOnLine`	`PointLiesOnLine(Point($),Line($1,$2))`
2	`PointLiesOnCircle`	`PointLiesOnCircle(Point($),Circle($))`
3	`Parallel`	`Parallel(Line($),Line($))`
4	`Perpendicular`	`Perpendicular(Line($),Line($))`
5	`IntersectAt`	`IntersectAt(Line($),Line($),Line($),Point($))`
6	`BisectsAngle`	`BisectsAngle(Line($),Angle($))`
7	`Congruent`	`Congruent(Polygon($),Polygon($))`
8	`Similar`	`Similar(Polygon($),Polygon($))`
9	`Tangent`	`Tangent(Line($),Circle($))`
10	`Secant`	`Secant(Line($),Circle($))`
11	`CircumscribedTo`	`CircumscribedTo(Shape($),Shape($))`
12	`InscribedIn`	`InscribedIn(Shape($),Shape($))`

Table 5.10: 12 predicates and corresponding literal templates for binary geometric relations.

#	Predicates	Literal templates
1	`IsMidpointOf`	`IsMidpointOf(Point($),Line($))`
2	`IsCentroidOf`	`IsCentroidOf(Point($),Shape($))`
3	`IsIncenterOf`	`IsIncenterOf(Point($),Shape($))`
4	`IsRadiusOf`	`IsRadiusOf(Line($),Circle($))`
5	`IsDiameterOf`	`IsDiameterOf(Line($),Circle($))`
6	`IsMidsegmentOf`	`IsMidsegmentOf(Line($),Triangle($))`
7	`IsChordOf`	`IsChordOf(Line($),Circle($))`
8	`IsSideOf`	`IsSideOf(Line($),Polygon($))`
9	`IsHypotenuseOf`	`IsHypotenuseOf(Line($),Triangle($))`
10	`IsPerpendicularBisectorOf`	`IsPerpendicularBisectorOf(Line($),Triangle($))`
11	`IsAltitudeOf`	`IsAltitudeOf(Line($),Triangle($))`
12	`IsMedianOf`	`IsMedianOf(Line($),Quadrilateral($))`
13	`IsBaseOf`	`IsBaseOf(Line($),Quadrilateral($))`
14	`IsDiagonalOf`	`IsDiagonalOf(Line($),Quadrilateral($))`
15	`IsLegOf`	`IsLegOf(Line($),Trapezoid($))`

Table 5.11: 15 predicates and corresponding literal templates for A-IsXOf-B-type geometric relations.

#	Predicates	Literal templates
1	`SinOf`	`SinOf(Var)`
2	`CosOf`	`CosOf(Var)`
3	`TanOf`	`TanOf(Var)`
4	`CotOf`	`CotOf(Var)`
5	`HalfOf`	`HalfOf(Var)`
6	`SquareOf`	`SquareOf(Var)`
7	`SqrtOf`	`SqrtOf(Var)`
8	`RatioOf`	`RatioOf(Var),RatioOf(Var1,Var2)`
9	`SumOf`	`SumOf(Var1,Var2,...)`
10	`AverageOf`	`AverageOf(Var1,Var2,...)`
11	`Add`	`Add(Var1,Var2,...)`
12	`Mul`	`Mul(Var1,Var2,...)`
13	`Sub`	`Sub(Var1,Var2,...)`
14	`Div`	`Div(Var1,Var2,...)`
15	`Pow`	`Pow(Var1,Var2)`
16	`Equals`	`Equals(Var1,Var2)`
17	`Find`	`Find(Var)`
18	`UseTheorem`	`UseTheorem(A_B_C)`

Table 5.12: 18 predicates and corresponding literal templates for numerical attributes and relations.

CHAPTER 6

Automatic Demonstration Learning

6.1 Introduction

Large Language Models (LLMs) such as GPT-3 [BMR20] have demonstrated remarkable abilities in in-context learning [WWS22b, WWS22a], where they can learn from the context of the data presented to them. For example, given a task description and a few examples, an LLM can be prompted to solve math word problems by following the provided chain-of-thought reasoning process, which consists of intermediate reasoning steps leading to the final output.

However, the performance of LLMs in in-context learning can be highly unstable, as it depends on the selection of context examples [ZWF21, LSZ22, LBM22], typically sourced from the training data. Research has shown that performance can fluctuate widely under different permutations of examples, ranging from random chance to perfect performance, indicating a high variance in LLM's accuracy. This instability is particularly evident in benchmarks like TabMWP [LQC23], a mathematical reasoning dataset with tabular contexts, where problems are distributed across multiple question types and diverse table layouts.

To address this challenge, we propose the task of automatic prompt learning, where the goal is to select optimal examples from a set of candidates to use as the prompt, given a test query. For instance, given a test query and five candidate examples, we aim to select the best two examples to form the prompt for the LLM, such as GPT-3. Selecting good examples enables the LLM predictions to be as stable and accurate as possible. However, the challenge lies in the vast search space, determined by the number of permutations. For example, selecting a 4-shot prompt from 100 candidate examples would result in P_{100}^{4} possibilities. Moreover, there is no ground truth available for the selection of in-context examples.

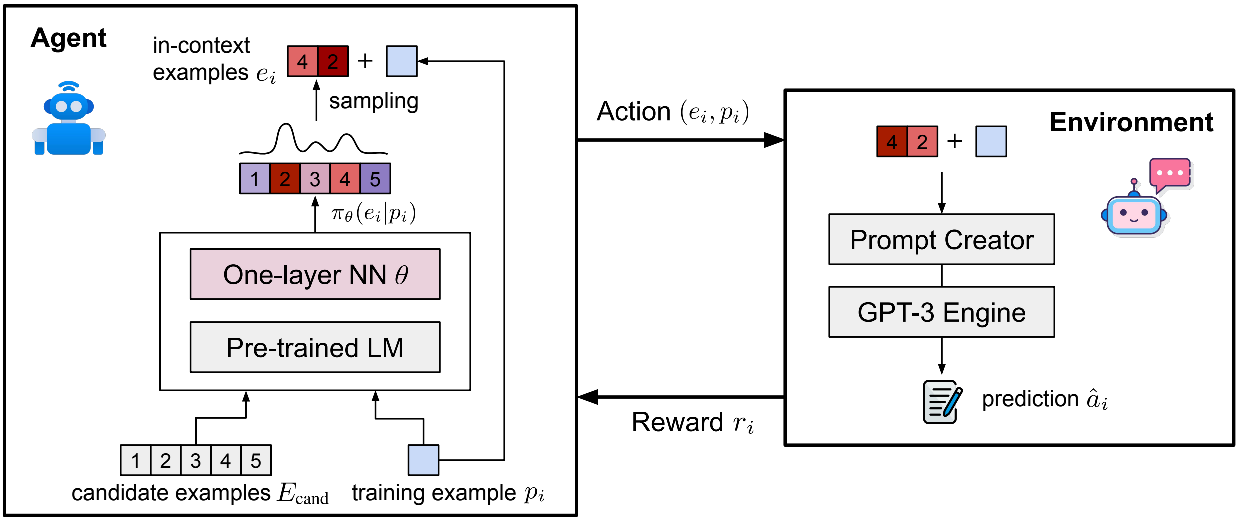


Figure 6.1: Overview of our proposed PromptPG approach, which learns to select high-performing in-context examples via policy gradient by interacting with the GPT-3 API, without relying on any manually designed heuristics.

To tackle this challenge, we propose PromptPG, a novel approach that learns to select in-context examples from a small amount of training data via policy gradient for prompt learning. As illustrated in Figure 6.1, an agent learns to find optimal in-context examples from a candidate pool, aiming to maximize the prediction rewards on given training examples when interacting with the GPT-3 environment. A policy network, built on top of the BERT language model [KT19] with fixed parameters and a one-layer linear neural network with learnable parameters, defines the strategy for selecting in-context examples given the current training example. The learnable parameters are updated following the policy gradient strategy [SB98]. Unlike random selection [WWS22b, WWS22a], brute-force search, or retrieval-based selection [LSZ22], PromptPG constructs the prompt dynamically given the candidate pool when interacting with the GPT-3 API.

We conduct extensive experiments with PromptPG and other baselines on TabMWP. We implement two state-of-the-art methods as baselines: UnifiedQA [KMK20] for general question answering and TAPEX [LCG22] for tabular question answering, both in pre-trained and fine-tuned settings. Experimental results show that PromptPG achieves an overall accuracy of 68.23% on TabMWP, surpassing previous methods by a large margin of up to 5.31%. Further analysis demonstrates that PromptPG selects better in-context examples compared to a wide range of existing selection strategies and significantly reduces prediction variance compared to random selection.

The main contributions of our work are as follows: (a) We propose PromptPG, a novel approach that learns the prompt dynamically via policy gradient to select in-context examples for few-shot GPT-3. To the best of our knowledge, it is the first work that applies reinforcement learning to select in-context examples for the few-shot GPT-3 model; (b) Experimental results show that PromptPG achieves an improvement of up to 5.31% on TabMWP over existing methods, with reduced selection instability compared to random selection.

6.2 Related Work

6.2.1 Prompt Learning for Language Models

Large pre-trained language models, such as GPT-3 [BMR20], have shown their remarkable ability of few-shot learning on a wide range of downstream tasks [HGJ19, BMR20, MYZ22, LMX22]. Given a few in-context examples as demonstrations, GPT-3 can generalize to unseen test examples without parameter updating. For example, [WWS22b] randomly select different in-context examples from the training set and formulate their corresponding prompt with a test sample. However, recent studies show that few-shot GPT-3 highly depends on the selection of in-context examples and could be unstable, varying from the near chance to near state-of-the-art performance [ZWF21, LSZ22, LQY23]. To mitigate the volatility of selecting in-context examples, [LBM22] propose retrieving relevant examples that are semantically similar to the test sample. Other possible strategies could be using brute-force permutation search or relying on manually designed heuristics like choosing the most complex examples. Inspired by reinforcement learning's ability to search for an optimal action policy, we propose applying the policy gradient strategy [SB98] to learn to select in-context examples more efficiently and stably without designing human-designed heuristics.

6.2.2 Policy Gradient

Policy gradient is an approach to solving reinforcement learning problems that target modeling and optimizing the policy directly. Many policy gradient algorithms have been proposed in the past decade [SLH14, LHP15, MBM16, SWD17, BHB18]. They have been proven effective in areas like robotics [PS06] and chatbots [KBT16]. In recent work that focuses on aligning language models with human values [OWJ22, QZL22, GMT22], policy gradient has been used to optimize language models with rewards learned from human feedback and preference. To the best of our knowledge, our PromptPG is the first work that proposes to select prompts dynamically for large pre-trained language models in the mathematical reasoning field.

6.3 Dynamic Prompting via Policy Gradient

The in-context examples can be randomly [WWS22b, WWS22a] or retrieval-based selected [LSZ22] from the training set. Recent research, however, has shown that few-shot GPT-3 can be highly unstable across different selections of in-context examples and permutations of those examples [ZWF21, LSZ22, LBM22]. This instability may be more severe on mathematical reasoning problems, where examples are more distinct because they include various question types and heterogeneous context formats. To alleviate this issue, we aim to propose a novel approach that can learn to select performing in-context examples using a policy gradient strategy, without brute-force searching or manually designed heuristics, as summarized in Algorithm 2.

Formally, given a TabMWP problem p_i, we want the agent to find K in-context examples $e_i = \{e_i^1, e_i^2, ..., e_i^K\}$ from a candidate pool E_{cand}, and generate the answer $\hat{a}_i$, maximizing a reward $r_i = R(\hat{a}_i|p_i)$. The in-context examples are selected according to a policy

$$e_i^k \sim \pi_\theta(e_i|p_i),\ e_i^k \in E_{\text{cand}}, e_i^k \text{ are independent for } k = \{1, 2, ..., K\}, \tag{6.1}$$

where θ are the policy's parameters. The answer is generated through: $\hat{a}_i = \text{GPT-3}(e_i, p_i)$ using the selected examples and the given problem as the input prompt. The reward is then computed by evaluating the generated answer $\hat{a}_i$ with respect to the ground truth answer a_i:

$$r_i = R(\hat{a}_i|p_i) = \textsc{Eval}(\hat{a}_i, a_i),\ r_i \in \{-1, 1\}. \tag{6.2}$$

The function $\textsc{Eval}()$ returns a reward of 1 if the generated answer aligned with the label and -1

otherwise. Our goal is to maximize the expected reward of the generated answer under the policy $\mathbb{E}_{e_i \sim \pi_\theta(e_i|p_i)}[R(\text{GPT-3}(e_i, p_i))]$. We optimize the reward with respect to the parameters of the policy network using the Policy Gradient method [SB98]. The expected reward cannot be computed in closed form, so we compute an unbiased estimation with Monte Carlo Sampling,

$$\mathbb{E}_{e_i \sim \pi_\theta(e_i|p_i)} \left[R(\text{GPT-3}(e_i, p_i))\right] \approx \frac{1}{N} \sum_{i=1}^{N} R(\text{GPT-3}(e_i, p_i)),\ e_i \sim \pi_\theta(e_i|p_i), \tag{6.3}$$

where N is the size of each batch yielded from our training problem set P_{train}. In this work, we experiment using the REINFORCE policy gradient algorithm [Wil92]:

$$\begin{aligned}\nabla \mathbb{E}_{e_i \sim \pi_\theta(e_i|p_i)} \left[R(\text{GPT-3}(e_i, p_i))\right] &= \mathbb{E}_{e_i \sim \pi_\theta(e_i|p_i)} \nabla_\theta \log(\pi_\theta(e_i|p_i)) R(\text{GPT-3}(e_i, p_i)) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log(\pi_\theta(e_i|p_i)) R(\text{GPT-3}(e_i, p_i)),\ e_i \sim \pi_\theta(e_i|p_i).\end{aligned} \tag{6.4}$$

Intuitively, if the predicted answer is correct, we update the policy so that the probability of selecting the same prompts gets higher. Otherwise, we update the policy to reduce the probability of selecting such less matched examples. The learning process is summarized in Algorithm 2 in the appendix.

To get the contextualized representation of the given problem and candidate examples, we use the BERT [KT19] `[CLS]` token representation as the problem encoding. We add a small linear layer on top of the BERT final pooling layer. That allows our model to learn both the semantic similarity that the pre-trained BERT model provides and the hidden logical similarity shared among the math problems. During training, the parameters of BERT are fixed and only the appended linear layer is updated, i.e., θ is composed of the learnable parameters $\mathbf{W}$ and $\mathbf{b}$:

$$\begin{aligned}\mathbf{h}(e_i) &= \mathbf{W}(\text{BERT}(e_i)) + \mathbf{b}, \\ \mathbf{h}(p_i) &= \mathbf{W}(\text{BERT}(p_i)) + \mathbf{b}, \\ \pi_\theta(e_i|p_i) &= \frac{\exp\left[\mathbf{h}(e_i) \cdot \mathbf{h}(p_i)\right]}{\sum_{e_i' \in E_{\text{cand}}} \exp\left[\mathbf{h}(e_i') \cdot \mathbf{h}(p_i)\right]}.\end{aligned} \tag{6.5}$$

Algorithm 2 Dynamic Prompt Learning via Policy Gradient (PromptPG)

Input: Initial policy π_{θ_0}, training example set P_{train}, candidate example set E_{cand}, # of training epochs N
Output: Learned policy π_θ

```
function REINFORCE($\pi_{\theta_0}$, $P_{\text{train}}$, $E_{\text{cand}}$, $N$)
    Initialize policy network $\pi$ with parameter $\theta_0$
    for epoch = $1, 2, ..., N$ do
        for $P_{\text{batch}} \in P_{\text{train}}$ do                              ▷ get a batch from the training set
            $\mathcal{L}_{\text{batch}} \leftarrow 0$
            for $p_i \in P_{\text{batch}}$ do
                Sample $e_i^k \sim \pi_\theta(e_i|p_i), e_i^k \in E_{\text{cand}}, k = \{1, ..., K\}$    ▷ $K$ is # of in-context examples
                $\hat{a}_i \leftarrow$ GPT-3$(e_i^1, ..., e_i^k, p_i)$              ▷ $\hat{a}_i$ is the GPT-3 generated answer
                $r_i \leftarrow \textsc{Eval}(\hat{a}_i, a_i), r_i \in \{-1, 1\}$     ▷ $a_i$ is the ground truth answer of $p_i$
                $\mathcal{L}_{\text{batch}} \leftarrow \mathcal{L}_{\text{batch}} - r_i \cdot \ln \pi_\theta(e_i|p_i)$
            end for
            Optimize $\mathcal{L}_{\text{batch}}$ wrt. $\theta$
        end for
    end for
    return $\pi_\theta$
end function
```

6.4 Experiments

6.4.1 Evaluation Task

We conduct extensive experiments with PromptPG and other baselines on TabMWP, a mathematical reasoning benchmark featuring tabular contexts. TabMWP includes diverse question types and various tabular formats. Each question is annotated with a step-by-step solution, making it an excellent benchmark for revealing performance fluctuations in LMs due to different selections of few-shot examples, and for verifying the effectiveness of PromptPG in selecting optimal few-shot examples from candidates.

A tabular math word problem p in TabMWP is represented as a pair (t, q), where t is a table context and q is a question. The table t could be represented in a visual format as an image, semi-structured text, or a structured database. In this work, we focus on the semi-structured format as the table context for simplicity. The table t features complicated layouts and formats: it contains multiple rows and columns, and each cell can be a string of text, a string of a number, or a mix of them. Depending on the question and answer types, the question q may be accompanied by multiple-choice options $c = \{c_1, c_2, \ldots, c_n\}$ or a unit u. Given a semi-structured tabular context t

and an unstructured question text q, the task is to generate the answer a, which is either numerical only text for a *free-text* question, or a text span from given options for a *multiple-choice* question.

6.4.2 Evaluation Baselines

We first develop two large language models, UnifiedQA [KMK20] and TAPEX [LCG22], in both pre-trained and fine-tuned settings, as strong baselines on TabMWP. Different model sizes are included to examine the performance across different model capacities. We further implement the zero-shot GPT-3 model, the few-shot GPT-3 model, and their chain-of-thought (CoT) reasoning variants [WWS22b]. We also study the heuristic guess baseline and human performance to analyze the lower and upper bounds on TabMWP, respectively.

Heuristics guess. To investigate the lower bound of the accuracy on TabMWP, we design simple heuristics to guess answers for each question type. For *multi-choice* questions, we randomly select one from the given options with even probabilities. For *free-text* questions on TabMWP, the answers could only be integral or decimal numbers. Intuitively, we take advantage of regular expressions to extract all the numbers from the tabular context and the question text as candidates, and then randomly choose one number as the prediction.

UnifiedQA baselines. UnifiedQA [KMK20] is a T5-based [RSR20] QA system that was pre-trained on 8 seed QA datasets of multiple formats but with a unified text-to-text paradigm. We load the pre-trained checkpoint as the pre-trained baseline and train it on TabMWP as the fine-tuned baseline. Three different parameter sizes are compared: SMALL (60M), BASE (220M), and LARGE (770M).

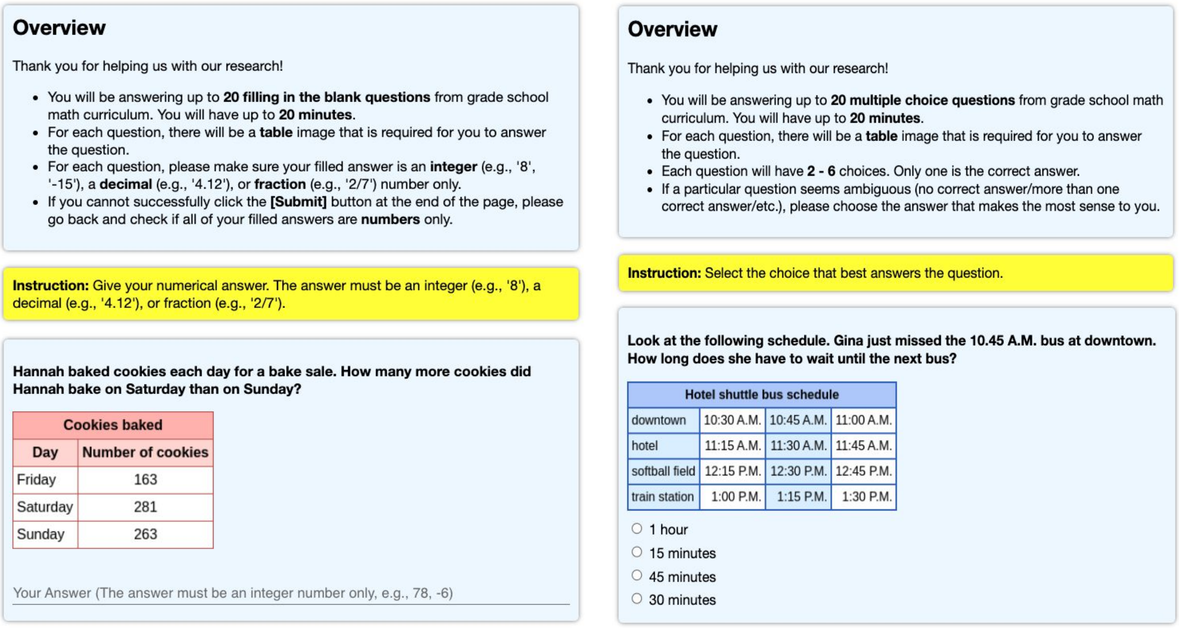
TAPEX baselines. TAPEX [LCG22] is a BART-based [LLG20] language model pre-trained on structured tabular data to mimic the behavior of a SQL executor that can answer table-based questions. TAPEX shows state-of-the-art performance on four table-related datasets. We establish the pre-trained and fine-tuned baselines on top of TAPEX with two model sizes: BASE (140M) and LARGE (400M).

Zero-shot GPT-3 and zero-shot-CoT GPT-3. We establish the zero-shot baseline based on GPT-3 [BMR20]. The zero-shot setup follows the format of TQ(C)$\rightarrow$A where the input is the concatenation of tokens of the tabular context (T), the question text (Q), and choice options (C) that apply while the output is to predict the answer (A). Following [KGR22], we further build zero-shot-CoT GPT-3, which refers to the GPT-3 model with a chain-of-thought (CoT) prompt. Specifically, we add the prompt "*Let's think step by step*" at the end of the input to ask the model to generate the multi-step solution (S) to mimic the reasoning process as humans. Then the model takes the raw input and the newly generated solution to predict the final answer.

Few-shot GPT-3 and few-shot-CoT GPT-3. Inspired by the recent progress achieved by GPT-3 in solving MWPs [WWS22b, WWS22a, KGR22], we evaluate TabMWP using GPT-3 models in few-shot learning settings. Provided with a few in-context examples of math word problems as the context, GPT-3 can generate the answer for a test problem and has shown impressive performance across different MWP datasets [WWS22b, WWS22a].

We follow the standard prompting [WWS22b] where in-context examples are randomly selected from the training data as demonstrations for the test example. Specifically, a few training examples, along with the test example p_i, are provided to GPT-3 for answer prediction. The prompt template is TQ(C)→SA, where each training example consists of a table context t, a question q, options c that apply, and an answer a. To generate the solution before the final answer, a solution s can be augmented in front of the answer a to reveal the multi-step reasoning process, which is able to boost the prediction performance [WWS22b]. To make the few-shot GPT-3 model workable on TabMWP, we utilize the semi-structured format as the tabular context.

Human study. To examine how humans perform on our TabMWP dataset, we released the human evaluation task on Amazon Mechanical Turk (AMT) to the test split. We designed two sub-tasks for the human study: answering the *free-text* questions and answering the *multi-choice* questions. The user interfaces for the two sub-tasks are shown in Figure 6.2. Each human intelligence task (HIT) contains 5 exam questions and 15 test questions. A worker should have a HIT Approval Rate of 98% or higher and be approved with 5,000 or more HITs. The worker is provided with detailed instructions at the beginning and needs to pass at least 3 *free-text* exam questions or 4 *multi-choice* exam questions to be qualified for the human study. Each HIT is assigned to two different workers. We assign a reward of \$0.80 and \$0.60 for one HIT of *free-text* and *multi-choice* sub-tasks, respectively. The human study results are available in Table 6.1.



Overview

Thank you for helping us with our research!

- You will be answering up to **20 filling in the blank questions** from grade school math curriculum. You will have up to **20 minutes**.
- For each question, there will be a **table** image that is required for you to answer the question.
- For each question, please make sure your filled answer is an **integer** (e.g., '8', '-15'), a **decimal** (e.g., '4.12'), or **fraction** (e.g., '2/7') number only.
- If you cannot successfully click the **[Submit]** button at the end of the page, please go back and check if all of your filled answers are **numbers** only.

Instruction: Give your numerical answer. The answer must be an integer (e.g., '8'), a decimal (e.g., '4.12'), or fraction (e.g., '2/7').

Hannah baked cookies each day for a bake sale. How many more cookies did Hannah bake on Saturday than on Sunday?

Cookies baked	
Day	Number of cookies
Friday	163
Saturday	281
Sunday	263

Your Answer (The answer must be an integer number only, e.g., 78, -6)

Overview

Thank you for helping us with our research!

- You will be answering up to **20 multiple choice questions** from grade school math curriculum. You will have up to **20 minutes**.
- For each question, there will be a **table** image that is required for you to answer the question.
- Each question will have **2 - 6** choices. Only one is the correct answer.
- If a particular question seems ambiguous (no correct answer/more than one correct answer/etc.), please choose the answer that makes the most sense to you.

Instruction: Select the choice that best answers the question.

Look at the following schedule. Gina just missed the 10.45 A.M. bus at downtown. How long does she have to wait until the next bus?

Hotel shuttle bus schedule			
downtown	10:30 A.M.	10:45 A.M.	11:00 A.M.
hotel	11:15 A.M.	11:30 A.M.	11:45 A.M.
softball field	12:15 P.M.	12:30 P.M.	12:45 P.M.
train station	1:00 P.M.	1:15 P.M.	1:30 P.M.

○ 1 hour
○ 15 minutes
○ 45 minutes
○ 30 minutes

Figure 6.2: User interfaces for the human study on the TabMWP dataset, showcasing the design for both *free-text* and *multi-choice* question types.

6.4.3 Evaluation metric

The answer part is extracted from the GPT-3 generation using manually designed regular expressions. To evaluate the baselines and our method, we utilize the accuracy metric to determine if the generated answer is correct given the ground truth answer. For *free-text* problems where the answer is set as a number, we normalize the prediction and the label to decimal numbers with two-digit precision and check if their values are equivalent. For *multi-choice* problems, we choose the most similar one from options to the generated answer following [KMK20].

6.4.4 Implementation details.

Fine-tuned UnifiedQA and TAPEX baselines are trained on the train split and evaluated on the test split. Few-shot GPT-3 and few-shot-CoT GPT-3 randomly select two in-context examples from the training data to build the prompt. Our PromptPG is built on top of few-shot GPT-3 with a different selection strategy: (a) in the training stage, the agent learns to select two examples from 20 candidates and is evaluated on 160 training examples to calculate the reward; (b) in the test stage, the agent with an optimal policy chooses two examples from 20 candidates for each test example. The candidates are randomly selected from the training set. Experiments for two few-shot GPT-3 baselines and our PromptPG are repeated three times, and the average accuracy is reported in Table 6.1.

Our experiments for UnifiedQA baselines, TAPEX baselines, and our proposed PromptPG are conducted using PyTorch on two Nvidia RTX 3090 GPUs. For fine-tuning the UnifiedQA and TAPEX baselines, we use the Adam optimizer [KB14] with an initial learning rate of $5\mathrm{e}{-5}$. The training process takes 10 epochs with a batch size of 16. The maximum number of input tokens is set as 200 and the maximum output length is 100.

In our proposed PromptPG, the embedding size of the added linear neural network is 768. To learn the policy network, we use the Adam optimizer with an initial learning rate of $1\mathrm{e}{-3}$. The maximum number of training epochs is 30, with a batch size of 20. The training process is stopped early if there is any NaN value in the loss for a batch of training data.

For the GPT-3 engine, we use TEXT-DAVINCI-002, the most capable engine recommended by the official documentation. The temperature is set as 0 and the top probability is set as 1.0 to get the most deterministic prediction. The maximum number of tokens allowed for generating text is 512. Both the frequency penalty and the presence penalty are set as the default value, i.e., 0.

6.4.5 Experimental Results

Table 6.1 demonstrates the results of different baselines and our method on the TabMWP dataset. Benefiting from pre-training on the tabular corpus, the TAPEX baseline performs better on average than UnifiedQA with a similar model size, which is only pre-trained on unstructured textual data. Increasing the model size can improve the prediction accuracy for both UnifiedQA and TAPEX. Fine-tuned on TabMWP, the baseline models can significantly improve the prediction performance on the average and all aggregated accuracy metrics.

Without any examples provided to GPT-3, zero-shot GPT-3 achieves a comparable accuracy to the best fine-tuned baselines, UnifiedQA$_{\text{LARGE}}$ and TAPEX$_{\text{LARGE}}$, showing its surprisingly good generalization ability on TabMWP. Provided with two randomly sampled in-context examples as the prompt, few-shot GPT-3 gets an improvement of 0.17%. Generating the multi-step solution

Method	Training Data	Selection Strategy	Question Types		Answer Types					Grades		Avg.
			FREE	MC	INT	DEC	EXTR	BOOL	OTH	1-6	7-8	
Heuristic Baselines												
Heuristic guess	-	-	6.71	39.81	8.37	0.26	30.80	51.22	26.67	17.55	12.27	15.29
Human performance	-	-	84.61	93.32	84.95	83.29	97.18	88.69	96.20	94.27	81.28	90.22
pre-trained Baselines												
UnifiedQA$_{\text{SMALL}}$	-	-	1.18	43.62	1.37	0.43	38.70	49.78	37.14	15.57	7.65	12.18
UnifiedQA$_{\text{BASE}}$	-	-	4.60	43.02	5.28	1.97	37.08	50.11	38.10	17.14	11.11	14.56
UnifiedQA$_{\text{LARGE}}$	-	-	4.48	48.80	5.19	1.72	48.33	50.33	40.00	19.78	10.87	15.96
TAPEX$_{\text{BASE}}$	-	-	7.32	39.76	8.68	2.06	35.06	47.11	20.95	18.67	11.81	15.73
TAPEX$_{\text{LARGE}}$	-	-	8.80	46.59	10.62	1.72	46.91	48.11	30.48	22.65	13.18	18.59
fine-tuned Baselines												
UnifiedQA$_{\text{SMALL}}$	23,059	-	22.27	51.31	27.27	2.83	52.28	48.11	69.52	35.85	21.71	29.79
UnifiedQA$_{\text{BASE}}$	23,059	-	34.02	70.68	40.74	7.90	84.09	55.67	73.33	53.31	30.46	43.52
UnifiedQA$_{\text{LARGE}}$	23,059	-	48.67	82.18	55.97	20.26	94.63	68.89	79.05	65.92	45.92	57.35
TAPEX$_{\text{BASE}}$	23,059	-	39.59	73.09	46.85	11.33	84.19	61.33	69.52	56.70	37.02	48.27
TAPEX$_{\text{LARGE}}$	23,059	-	51.00	80.02	59.92	16.31	**95.34**	64.00	73.33	67.11	47.07	58.52
Prompting Baselines w/ GPT-3												
Zero-shot	-	-	53.57	66.67	55.55	45.84	78.22	55.44	54.29	63.37	48.41	56.96
Zero-shot-CoT	-	-	54.36	66.92	55.82	48.67	78.82	55.67	51.43	63.62	49.59	57.61
Few-shot (2-shot)	2	Random	54.69	64.11	58.36	40.40	75.95	52.41	53.02	63.10	49.16	57.13
Few-shot-CoT (2-shot)	2	Random	60.76	69.09	60.04	63.58	76.49	61.19	**67.30**	68.62	55.31	62.92
PromptPG *w/ GPT-3 (Ours)*												
Few-shot-CoT (2-shot)	160+20	Dynamic	**66.17**	**74.11**	**64.12**	**74.16**	76.19	**72.81**	65.71	**71.20**	**64.27**	$\mathbf{68.23}_{5.31\uparrow}$

Table 6.1: Evaluation results of various baselines and our method on TabMWP.

before the answer, the few-shot-CoT GPT-3 model reports the best performance among all of these baseline models, with an accuracy of 62.92%. Unlike few-shot-CoT GPT-3 randomly selecting the in-context examples, our proposed PromptPG learns to select performing examples with the help of policy gradient. PromptPG establishes a state-of-the-art performance on the TabMWP dataset: it surpasses the best baseline few-shot-CoT GPT-3 by 5.31% on average. PromptPG shows its consistent advantages on two question types, two grade groups, and most of the answer types.

Heuristic guess and human performance. The accuracy of *multi-choice* questions by heuristic guess is 39.81%, which aligns with the fact that there are 2.88 options on average. The accuracy for *free-text* questions is considerably low since the inputs of TabMWP problems do not have direct clues for the answers. Humans outperform all benchmarks consistently across question types, answer types, and grade groups, with a 21.99% average accuracy advantage over our best performing PromptPG. This gap is to be filled by future research on semi-structured mathematical reasoning.

Problem types and difficulty. Among all the baselines, we find it is easier for models to answer *multi-choice* questions than *free-text* questions. Questions with the boolean (BOOL) and other (OTH) answer types tend to have lower accuracy scores than the extractive (EXTR) answer type, because the former ones need the abilities of fact verification and language understanding on diverse options, respectively. It is also not surprising for us to find that all the models perform worse on problems in grades 7-8 than in a lower-level group of 1-6.

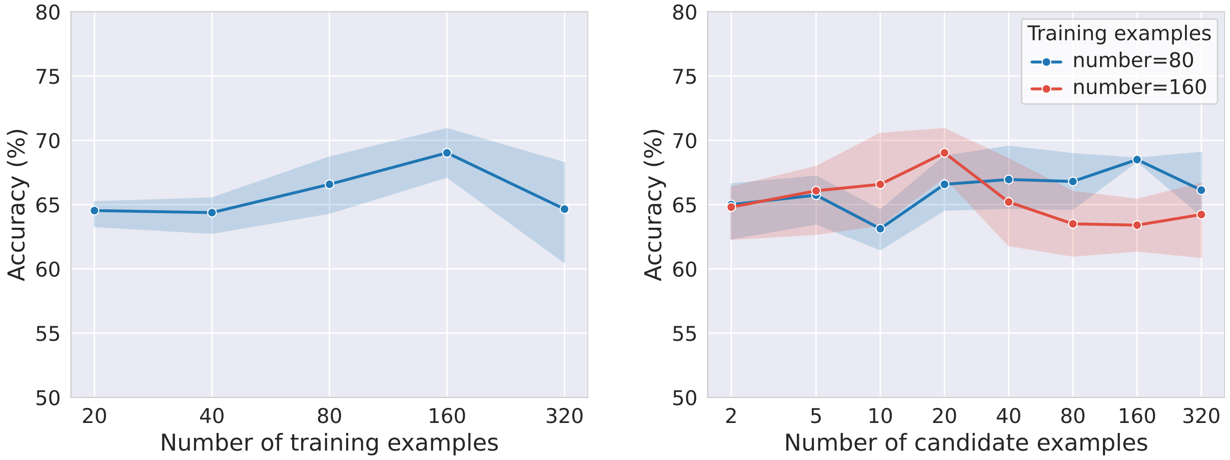
6.4.6 Ablation Study

Here, we will study how different factors have an effect on the performances of baselines and our method on TabMWP. Experiments are conducted on 1,000 development examples.

Blind study of the dataset. We evaluate the information gain of each component of the TabMWP problems by removing it from model inputs. To eliminate the impact and variance caused by example selection, the study is conducted using the zero-shot GPT-3 model. As shown in Table 6.2, there is a dramatic decline when either the tabular context (T) or the question text (Q) is missing from the inputs. For example, T→A and Q→A only attain an average accuracy of 6.10% and 7.00%, respectively, and their accuracies are near to zero on the *multi-choice* questions. Taking both tabular and textual data as inputs (TQ→A), the model significantly beats the heuristic guess. With the complete input information (TQ(C)→A), the full model achieves the best performance. The blind study shows that our TabMWP is robust and reliable in distribution, and all input components are indispensable parts that provide necessary information for answering the questions.

Model	**Format**	**FREE**	**MC**	**INT**	**DEC**	**EXTR**	**BOOL**	**OTH**	**1-6**	**7-8**	**Avg.**
Heuristic guess	TQ(C)→A	7.31	40.36	9.20	0.00	34.44	47.32	50.00	17.99	13.96	16.40
Zero-shot GPT-3	T→A	8.28	0.36	10.24	0.67	0.66	0.00	0.00	9.41	1.02	6.10
Zero-shot GPT-3	Q→A	9.24	1.09	10.94	2.68	1.32	0.89	0.00	10.23	2.03	7.00
Zero-shot GPT-3	T(C)→A	8.28	41.82	10.24	0.67	36.42	50.89	25.00	23.60	8.12	17.50
Zero-shot GPT-3	Q(C)→A	9.10	33.09	10.94	2.01	25.17	44.64	25.00	21.29	7.11	15.70
Zero-shot GPT-3	TQ→A	55.31	68.36	56.60	50.34	79.47	54.46	58.33	66.34	47.46	58.90
Zero-shot GPT-3 (full model)	TQ(C)→A	54.76	72.00	56.42	48.32	76.82	66.07	66.67	67.00	47.97	59.50

Table 6.2: Blind studies on TabMWP. T: tabular context; Q: question; C: choice options; A: answer. Q(C) means choice options follow the question in the input, while Q refers to the question only.



(a) Accuracy w.r.t. different numbers of training examples, given 20 candidate examples.

(b) Accuracy w.r.t. different numbers of candidates, given 80 and 160 training examples.

Figure 6.3: Accuracy of PromptPG with respect to different numbers of training and candidate examples. Experiments are conducted on 1,000 development instances from TabMWP, with each setting repeated using four random seeds to ensure robustness.

Number of training examples. We study the effect of different numbers of training examples on our dynamic prompt learning in Figure 6.3 (a). With more training examples, the prediction accuracy first gradually increases to a peak of around 160 training examples. After that, the accuracy goes down with a growing variance. We reckon it is because the policy gradient algorithm can benefit from the scaling-up training data but fails to exploit more examples efficiently.

Number of candidate examples. In Figure 6.3 (b), we investigate how different numbers of candidate examples can affect policy learning performance. With the increasing candidate number, it is observed that the prediction accuracy will first go up and then go down after a threshold, given 80 or 160 training examples. It is probably because when the candidate pool is too small, the policy gradient algorithm has a limited action space to explore enough problem types. In contrast, too many candidates could make the algorithm hard to learn an optimal policy in a large search space.

Different selection strategies. In Table 6.3, we compare the proposed PromptPG with random selection and other heuristic-based example selection strategies for the few-shot-CoT GPT-3 model. Compared to random selection, selecting the same question or answer type of examples helps the model to take the task-relevant examples as the prompt, thus improving the accuracy and reducing the variance. Choosing the most complex examples does not boost the prediction performance consistently. Manual selection selects the two examples from 20 with the highest

Selection strategy	**Acc. (%)**
Same question type	66.2 ± 0.60
Same answer type	67.9 ± 0.38
Same grade level	67.9 ± 1.87
Most complex (# of table cells)	64.0 ± 0.42
Most complex (# of ques. words)	68.2 ± 0.26
Random selection	65.2 ± 4.01
Manual selection (fixed w/ top 2)	66.9 ± 0.00
Nearest neighbor	68.2 ± 0.29
PromptPG (Ours)	**70.9 ± 1.27**

Table 6.3: Evaluation results of different selection strategies with three trials.

evaluation accuracy on one-shot-CoT GPT-3 as the fixed set of in-context examples. Although it achieves the lowest prediction variance of 0, it only improves by 1.7% over random selection. The most semantically similar examples, as a kind of nearest neighbor search of the test example, help construct the performing and stable prompt for GPT-3. PromptPG shows its effectiveness in selecting optimal in-context examples over other strategies and largely reduces the instability.

Number of few-shot examples. We study the few-shot-CoT GPT-3 model with random selection in terms of the different numbers of in-context shots. For each number of in-context shots, the experiment was conducted on 1,000 development examples and repeated three times. The results are shown in Table 6.4. When increasing the number of in-context shots from the current 2 to 4, the few-shot-CoT GPT-3 model reduces the prediction variance from the random selection of in-context shots and achieves an accuracy improvement of 2.5%. When the number of in-context shots is increased to 5, the model with random selection does not gain further benefits. Our PromptPG displays impressive advantages over random selection in terms of data efficiency and prediction accuracy. With only two in-context shots, PromptPG achieves the highest accuracy of 70.9% and a comparable low deviation compared to random selection with more shots.

6.4.7 Case Study

We conduct the case study of our proposed PromptPG. We visualize the two in-context examples selected by strategies of our PromptPG, nearest neighbor search, and random selection, in Figure

Model	**Selection strategy**	**Shot number**	**Acc. (%)**
Few-shot-CoT GPT-3	Random selection	2	65.2 ± 4.01
Few-shot-CoT GPT-3	Random selection	3	65.7 ± 1.16
Few-shot-CoT GPT-3	Random selection	4	67.7 ± 0.78
Few-shot-CoT GPT-3	Random selection	5	67.5 ± 0.98
Few-shot-CoT GPT-3	**PromptPG (ours)**	2	$\mathbf{70.9 \pm 1.27}$

Table 6.4: Results of different numbers of few-shot examples on 1,000 development examples.

6.4, 6.5, and 6.6, respectively.

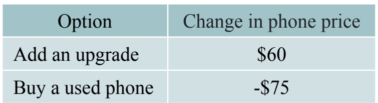
The nearest neighbor search strategy selects the "superficially" similar examples to the test example. It selects the most similar candidates to the test query based on semantic similarity, such as the BERT score of two sentences. For example, consider a test query asking, "How many more children than parents are playing tag?" Nearest neighbor search might select an example with similar wording like "How many children" in the question. However, despite the similar words, these two questions involve different reasoning steps. Therefore, this retrieved few-shot example does not effectively guide GPT-3 to generate the correct steps to answer the test question.

Instead, our PromptPG algorithm prefers examples that demonstrate multiple reasoning steps and showcase similar mathematical reasoning abilities to the test example. Given the same test query, our algorithm retrieves a different in-context example. The solution to this example first analyzes the question and offers a general solution. It then writes an expression to calculate the answer and finally concludes by providing the answer. With this example used as the prompt, GPT-3 follows similar steps to solve the test example and outputs the correct answer.

▷ *In-context example 1 (ID: 28463)*

Table:

Option | Change in phone price
Add an upgrade | $60
Buy a used phone | -$75



Option	Change in phone price
Add an upgrade	$60
Buy a used phone	-$75

Question: Luna is looking at the price of new cell phones online. Her favorite company, OrangeTech, has a special this weekend. Luna can add an upgrade to a phone for an additional cost, or she can buy a used phone to get a discount. The change in price for each option is shown in the table. Which option results in a greater change in price?

Options: (A) adding an upgrade (B) buying a used phone

Answer:

(Step 1) To find the option that results in a greater change in price, use absolute value. Absolute value tells you how much the price changes.

(Step 2) Add an upgrade: |$60| = $60

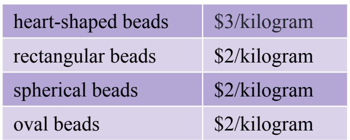
(Step 3) Buy a used phone: |-$75| = $75

(Step 4) Buying a used phone results in a greater change in price. It reduces the price by $75. The answer is buying a used phone.

▷ *In-context example 2 (ID: 13974)*

Table:

heart-shaped beads | $3/kilogram
rectangular beads | $2/kilogram
spherical beads | $2/kilogram
oval beads | $2/kilogram



heart-shaped beads	$3/kilogram
rectangular beads	$2/kilogram
spherical beads	$2/kilogram
oval beads	$2/kilogram

Question: Rebecca bought 2.5 kilograms of oval beads. How much did she spend? (Unit: $)

Answer:

(Step 1) Find the cost of the oval beads. Multiply the price per kilogram by the number of kilograms.

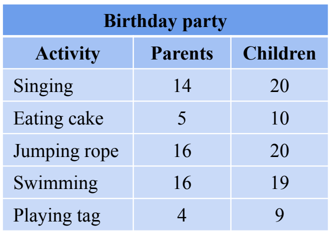
(Step 2) $2 × 2.5 = $5

(Step 3) She spent $5. The answer is 5.

▷ *Test example (ID: 17417)*

Table:

[TITLE]: Birthday party
Activity | Parents | Children
Singing | 14 | 20
Eating cake | 5 | 10
Jumping rope | 16 | 20
Swimming | 16 | 19
Playing tag | 4 | 9



Birthday party

Activity	Parents	Children
Singing	14	20
Eating cake	5	10
Jumping rope	16	20
Swimming	16	19
Playing tag	4	9

Question: At Josie's birthday party, children and parents celebrated by participating in some activities. How many more children than parents are playing tag? (Unit: children)

Answer:

(Step 1) To find the difference between the number of children and parents playing tag, subtract the number of parents from the number of children.

(Step 2) 9 - 4 = 5

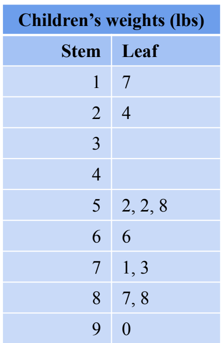
(Step 3) There are 5 more children than parents playing tag. The answer is 5.

Output: 5

Figure 6.4: Two in-context examples selected by PromptPG, the prompt, and the correct prediction. The selected examples require similar abilities of mathematical reasoning to the test example.

▷ *In-context example 1 (ID: 18429)*

Table:
[TITLE]: Children’s weights (lbs)
Stem | Leaf
1 | 7
2 | 4
3 |
4 |
5 | 2, 2, 8
6 | 6
7 | 1, 3
8 | 7, 8
9 | 0

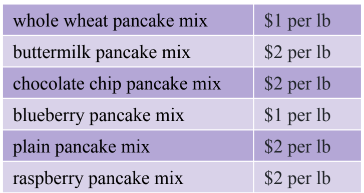

Children’s weights (lbs)

Stem	Leaf
1	7
2	4
3	
4	
5	2, 2, 8
6	6
7	1, 3
8	7, 8
9	0

Question: Dr. Thornton, a pediatrician, weighed all the children who recently visited her office. How many children weighed exactly 52 pounds? (Unit: children)
Answer:
(Step 1) For the number 52, the stem is 5, and the leaf is 2. Find the row where the stem is 5. In that row, count all the leaves equal to 2.
(Step 2) You counted 2 leaves, which are blue in the stem-and-leaf plot above.
(Step 3) 2 children weighed exactly 52 pounds. The answer is 2.

▷ *In-context example 2 (ID: 18129)*

Table:
whole wheat pancake mix | $1/lb
buttermilk pancake mix | $2/lb
chocolate chip pancake mix | $2/lb
blueberry pancake mix | $1/lb
plain pancake mix | $2/lb
raspberry pancake mix | $2/lb

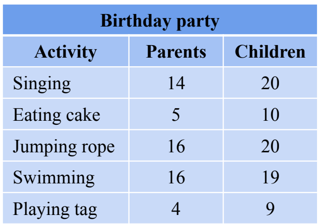


whole wheat pancake mix	$1 per lb
buttermilk pancake mix	$2 per lb
chocolate chip pancake mix	$2 per lb
blueberry pancake mix	$1 per lb
plain pancake mix	$2 per lb
raspberry pancake mix	$2 per lb

Question: If Vince buys 3.4 pounds of buttermilk pancake mix, how much will he spend? (Unit: $)
Answer:
(Step 1) Find the cost of the buttermilk pancake mix. Multiply the price per pound by the number of pounds.
(Step 2) $2 × 3.4 = $6.80
(Step 3) He will spend $6.80. The answer is 6.80.

▷ *Test example (ID: 17417)*

Table:
[TITLE]: Birthday party
Activity | Parents | Children
Singing | 14 | 20
Eating cake | 5 | 10
Jumping rope | 16 | 20
Swimming | 16 | 19
Playing tag | 4 | 9


Birthday party

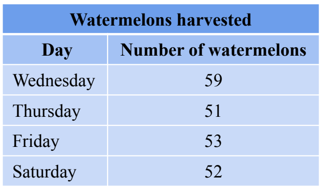
Activity	Parents	Children
Singing	14	20
Eating cake	5	10
Jumping rope	16	20
Swimming	16	19
Playing tag	4	9

Question: At Josie’s birthday party, children and parents celebrated by participating in some activities. How many more children than parents are playing tag? (Unit: children)
Answer:
(Step 1) For the activity “Playing tag”, the stem is 4, and the leaf is 9. Find the row where the stem is 4. In that row, count all the leaves equal to 9.
(Step 2) You counted 9 leaves, which are blue in the stem-and-leaf plot above. 9 children are playing tag. The answer is 9.
Output: 9
Ground truth: 5

Figure 6.5: In-context examples selected by *nearest neighbor search*, the prompt, and the incorrect prediction. The selected examples are semantically similar to the test example.

▷ *In-context example 1 (ID: 13033)*

Table:
[TITLE]: Watermelons harvested
Day | Number of watermelons
Wednesday | 59
Thursday | 51
Friday | 53
Saturday | 52

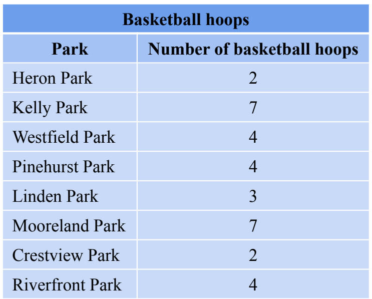


Watermelons harvested	
Day	**Number of watermelons**
Wednesday	59
Thursday	51
Friday	53
Saturday	52

Question: A farmer recalled how many watermelons were harvested in the past 4 days. On which day were the most watermelons harvested?
Options: (A) Wednesday (B) Thursday (C) Friday (D) Saturday
Answer:
(Step 1) Find the greatest number in the table. Remember to compare the numbers starting with the highest place value. The greatest number is 59.
(Step 2) Now find the corresponding day. Wednesday corresponds to 59. The answer is Wednesday.

▷ *In-context example 2 (ID: 32386)*

Table:
[TITLE]: Basketball hoops
Park | Number of basketball hoops
Heron Park | 2
Kelly Park | 7
Westfield Park | 4
Pinehurst Park | 4
Linden Park | 3
Mooreland Park | 7
Crestview Park | 2
Riverfront Park | 4

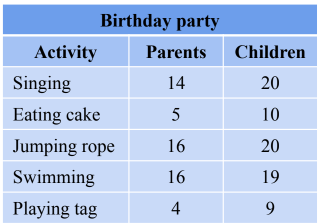


Basketball hoops	
Park	**Number of basketball hoops**
Heron Park	2
Kelly Park	7
Westfield Park	4
Pinehurst Park	4
Linden Park	3
Mooreland Park	7
Crestview Park	2
Riverfront Park	4

Question: The parks department compared how many basketball hoops there are at each park. What is the mode of the numbers?
Answer:
(Step 1) Read the numbers from the table.
(Step 2) First, arrange the numbers from least to greatest: 2, 2, 3, 4, 4, 4, 7, 7.
(Step 3) Now count how many times each number appears.
(Step 4) 2 appears 2 times. 3 appears 1 time. 4 appears 3 times. 7 appears 2 times.
(Step 5) The number that appears most often is 4.
(Step 6) The mode is 4. The answer is 4.

▷ *Test example (ID: 17417)*

Table:
[TITLE]: Birthday party
Activity | Parents | Children
Singing | 14 | 20
Eating cake | 5 | 10
Jumping rope | 16 | 20
Swimming | 16 | 19
Playing tag | 4 | 9



Birthday party		
Activity	**Parents**	**Children**
Singing	14	20
Eating cake	5	10
Jumping rope	16	20
Swimming	16	19
Playing tag	4	9

Question: At Josie's birthday party, children and parents celebrated by participating in some activities. How many more children than parents are playing tag? (Unit: children)
Answer:
(Step 1) Find the difference between the number of children and the number of parents for each activity.
(Step 2) For singing, the difference is 6.
(Step 3) For eating cake, the difference is 5.
(Step 4) For jumping rope, the difference is 4.
(Step 5) For swimming, the difference is 3.
(Step 6) For playing tag, the difference is 5.
(Step 7) The greatest difference is 6. The answer is 6.
Output: 6
Ground truth: 5

Figure 6.6: *Eandomly* selected in-context examples, the prompt, and the incorrect prediction. The selected examples have limited relevance to the test example,.

CHAPTER 7

Tool-Augmented Compositional Reasoning

7.1 Introduction

Remarkable progress has been observed in recent large language models (LLMs) for various natural language processing tasks, with prominent examples such as GPT-3 [BMR20], PaLM [CND23], LLaMA [ZHZ24], ChatGPT [Ope22], and the recently developed GPT-4 [Ope23a]. LLMs have demonstrated emergent abilities, including in-context learning and chain-of-thought (CoT) reasoning [WTB22]. These models are capable of solving diverse tasks in a zero-shot fashion [KGR22] or with the aid of a few examples [WWS22b], and they show great potential in planning and decision-making akin to human beings [HXX23, HAP22]. Despite these capabilities, LLMs face inherent limitations, such as an inability to access up-to-date information [KSW22], perform precise mathematical reasoning [PBG21, LQY23], or utilize specialized models [SDD23]. Therefore, enhancing current LLMs with the capability to automatically *compose* external tools for real-world task solving is critical to address these drawbacks.

Consider the example ② in Figure 7.1: *Which is the main persuasive appeal used in this ad?*. To answer this question, one needs to: 1) infer that there is an ad image containing text context and call a text decoder to understand the semantics; 2) retrieve background knowledge about *persuasive appeals* and the differences among three persuasive appeals; 3) generate a solution based on the input query and intermediate results from previous steps; and 4) finally produce the answer in a task-specific format. On the other hand, when answering *Which animal's skin is adapted for survival in cold places* (③), one might need to call modules such as an image captioner to decipher image information and a web search engine to retrieve domain knowledge to understand scientific terminologies. However, current tool-augmented LLMs still face challenges when addressing these

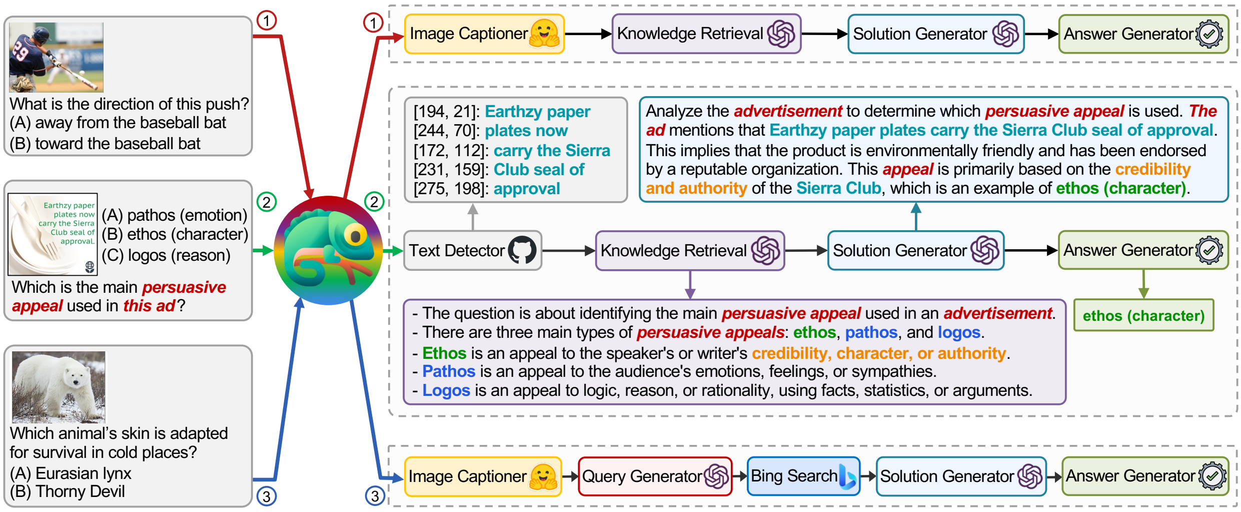

What is the direction of this push?
(A) away from the baseball bat
(B) toward the baseball bat
(A) pathos (emotion)
(B) ethos (character)
(C) logos (reason)
Which is the main persuasive appeal used in this ad?
Which animal's skin is adapted for survival in cold places?
(A) Eurasian lynx
(B) Thorny Devil
①
②
③
Image Captioner
Knowledge Retrieval
Solution Generator
Answer Generator
[194, 21]: Earthzy paper
[244, 70]: plates now
[172, 112]: carry the Sierra
[231, 159]: Club seal of
[275, 198]: approval
Analyze the advertisement to determine which persuasive appeal is used. The ad mentions that Earthzy paper plates carry the Sierra Club seal of approval. This implies that the product is environmentally friendly and has been endorsed by a reputable organization. This appeal is primarily based on the credibility and authority of the Sierra Club, which is an example of ethos (character).
Text Detector
Knowledge Retrieval
Solution Generator
Answer Generator
ethos (character)
- The question is about identifying the main persuasive appeal used in an advertisement.
- There are three main types of persuasive appeals: ethos, pathos, and logos.
- Ethos is an appeal to the speaker's or writer's credibility, character, or authority.
- Pathos is an appeal to the audience's emotions, feelings, or sympathies.
- Logos is an appeal to logic, reason, or rationality, using facts, statistics, or arguments.
Image Captioner
Query Generator
Bing Search
Solution Generator
Answer Generator

Figure 7.1: Examples of our Chameleon approach with GPT-4 on ScienceQA, a multi-modal question answering benchmark in scientific domains. Chameleon adapts to different queries by synthesizing programs that compose various tools and execute them sequentially to generate final answers.

real-world queries across various scenarios. Most existing approaches are either limited to a small number of tools [MFL22, CMW23, WLJ22, IDS23, PLS23, SDD23] or relying on domain-specific tools [NHB21, YLW23b, GK23, WYQ23, SMV23], and thus are not easy to generalize to queries of new domains. In this work, we study how to enable LLMs to synthesize programs to capture the logic of composing heterogeneous tools.

To address the challenges of existing work, we introduce Chameleon, a *plug-and-play compositional* reasoning framework that leverages LLMs to synthesize programs and compose various tools for a wide range of tasks. Unlike existing tool-augmented LLMs [SDD23, NHB21, YLW23b, GK23, WYQ23, SMV23], Chameleon uses a richer set of tools, including LLMs, off-the-shelf vision models, web search engines, Python functions, and heuristics-based modules. Moreover, Chameleon leverages the in-context learning capabilities of LLMs and builds on an LLM as a natural language planner, without requiring any training or carefully curated rules. Prompted by tool descriptions and usage examples, the planner infers a program composed of a sequence of tools to execute in order to generate the final response for a user query. Instead of generating programs in domain-specific languages [NHB21, SMV23, GK23], Chameleon generates natural-language-like (NL) programs (e.g., `[Text_Detector, Knowledge_Retrieval, Solution_Generator, Answer_Generator]` for the second query in Figure 7.1). The

NL-like programs are easy to understand and debug by users with limited programming experience, and easily extendable to new modules. During each module's execution, the module processes the query and cached context, returns a result determined by the module itself, and updates the query and context for subsequent execution. Composing modules as a sequential program allows subsequent modules to leverage prior cached context and updated queries.

We showcase the adaptability and effectiveness of Chameleon on the ScienceQA [LMX22] and TabMWP [LQC23] tasks. ScienceQA is a multi-modal question answering benchmark spanning multiple context formats and various scientific topics, while TabMWP is a mathematical benchmark involving diverse tabular contexts. These two benchmarks serve as a good testbed to evaluate Chameleon's ability to coordinate diverse tools across different types and domains. Notably, Chameleon with GPT-4 achieves an 86.54% accuracy on ScienceQA, significantly improving upon the best published few-shot model by 11.37%. On TabMWP, using GPT-4 as the underlying LLM, Chameleon achieves an improvement of 7.97% over chain-of-thought (CoT) prompted GPT-4 [WWS22b] and a 17.0% increase over the best-published model [CMW23], lifting the state of the art to 98.78%. Further studies suggest that using GPT-4 as a planner exhibits more consistent and rational tool selection and is able to infer potential constraints given the instructions, compared to other LLMs like ChatGPT.

Our contributions are as follows: (1) We develop a plug-and-play compositional reasoning framework, Chameleon, that effectively composes external tools to address inherent limitations of LLMs and tackle a broad range of reasoning tasks. (2) Relying on an LLM as a natural language planner to generate programs, Chameleon successfully integrates various tools, including LLMs, off-the-shelf vision models, web search engines, Python functions, and rule-based modules, to build a versatile and adaptable AI system capable of answering real-world queries. (3) We demonstrate Chameleon's effectiveness on two challenging benchmarks, significantly surpassing the state of the art.

7.2 Related Work

Compositional reasoning. Neural modular and compositional approaches have been explored to automatically perform desired sub-task decomposition, enhancing interpretability and adaptability across various reasoning tasks. Early work [ARD16a, ARD16b] posits that complex reasoning tasks are fundamentally compositional and proposes neural module networks (NMN) to decompose them into subtasks. However, these methods rely on brittle off-the-shelf parsers and are limited by module configurations. Some later work [JHV17, HAR17, HAD18, KKR21], takes a step further by predicting instance-specific network layouts in an end-to-end manner, without relying on parsers, using reinforcement learning [Wil92] and weak supervised learning.

In visual reasoning, models comprising a program generator and an execution engine have been proposed to combine deep representation learning and symbolic program execution [JHV17, YWG18]. In the domain of mathematical reasoning, an interpretable solver has been developed to incorporate theorem knowledge as conditional rules and perform symbolic reasoning step by step [LGJ21]. Our work takes inspiration from neural module networks, yet it offers several distinct advantages. First, Chameleon does not require expensive supervision of task-specific programs for modeling training. Instead, it generates sequential programs, consisting of modules, that are easy to generalize to various domains and tasks, allowing the extension to new modules in a plug-and-play manner. Second, Chameleon does not require any training, but uses the in-context learning capabilities of LLMs to generate programs prompted by natural language instruction and demonstrations.

Tool-augmented LLMs. In recent years, the development of large language models (LLMs) [SFA22, CND23, CHL24, TLI23, BMR20, Ope22, Ope23a] has made tremendous progress and has stimulated research in prompt learning [WWS22b, LQC23, KTF23] and instruction learning [TLI23, ZHZ24, PLH23, GHZ23]. Despite the impressive performance of LLMs, they suffer from inherent limitations, such as the inability to access up-to-date information [KSW22], utilize external tools [SDD23], or perform precise mathematical reasoning [PBG21, LQY23]. Recent benchmarks, such as ScienceQA and TabMWP [LMX22, LQC23, CYK23, CYK24, SHZ24, LBX24],

Model	**Tool Use**						**Skill Dimension**					**Inference & Extension**		
	Size	֍	🤗	🐙	b	🐍	Image	Web	Know.	Math	Table	Composition	Planning	Plug-n-Play
CoT [WWS22b]	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Lila [MFL22]	1	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
PoT [CMW23]	2	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
Code4Struct [WLJ22]	1	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
PAL [GMZ23]	2	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
MathPrompter [IDS23]	2	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
ART [PLS23]	4	✓	✗	✗	✓	✓	✗	✓	✗	✓	✗	✓	✗	✓
Toolformer [SDD23]	5	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	natural lang.	✗
WebGPT [NHB21]	10	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	program	✗
MM-ReAct [YLW23b]	10	✓	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	word match	✓
Visual ChatGPT [WYQ23]	10+	✓	-	-	✗	✗	✓	✗	✗	✗	✗	✓	natural lang.	✓
ViperGPT [SMV23]	10+	✓	-	-	✗	✗	✓	✗	✓	✓	✗	✓	program	✓
VisProg [GK23]	10+	✓	-	-	✗	✓	✓	✗	✗	✗	✗	✓	program	✓
HuggingGPT [SST23]	10+	✓	✓	✗	✗	✗	✓	✗	-	✗	-	✓	natural lang.	✓
Chameleon (ours)	10+	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	natural lang.	✓

Table 7.1: A comparison of work that augments large language models with tool usage.

have emerged to evaluate the capability of LLMs to tackle intricate reasoning challenges, especially those emphasizing the use of external tools. Concurrently, there has been a growing interest in harnessing external tools and modular approaches to augment LLMs. These augmented LLMs can access real-time information aided by web search engines [NHB21] and leverage domain-specific knowledge from external resources [YIW23]. Some work leverages the Python interpreter to generate complex programs to employ powerful computational resources, and execute logical reasoning tasks more effectively [WLJ22, GMZ23, CMW23, MFL22, IDS23, PLS23, LLC23]. For example, Toolformer [SDD23] constructs tool-use augmented data to train language models to select five tools. In the realm of visual tools, various approaches have been proposed to enhance the capabilities of large language models in handling visual tasks [YLW23b, WYQ23, SMV23, GK23, SST23], augmented with Hugging Face models [SST23], Azure models [YLW23b], visual foundation models [WYQ23].

We compare Chameleon with other tool-augmented language models in Table 7.1. Many of these approaches are either constrained to a small set of tools or limited to task-specific tools, which reduces their capabilities across various skill dimensions and hampers their generalizability to new tasks. A recent line of work relies on large amounts of supervision [SDD23, KSW22] and focuses on generating commands [NHB21] and programs [SMV23, GK23] to infer the choice of tools. However, this approach needs to carefully tailored prompts to specific tasks and particular tools, and is neither flexible nor adaptive. In contrast, Chameleon instructs LLMs with natural language instructions that simply describe the roles of each module and provide a few calling examples, eliminating the need for additional training or tool-specific prompts when learning to compose different tools. More importantly, Chameleon offers users flexibility in terms of tool types and sources, updating the underlying LLMs, adding new tools, and adapting to new tasks. Our work shares the same spirit of AutoGPT [Ric23], an autonomous GPT-4 agent with the artificial general intelligence (AGI) ambition to incorporate numerous tools to achieve user-defined goals. While AutoGPT is still under development, our work is the first to instantiate the idea and verify its effectiveness on well-studied benchmarks.

7.3 General Framework: Chameleon

To address the limitations of current LLMs in utilizing diverse tools, we propose Chameleon, a novel *plug-and-play compositional* reasoning framework, synthesizing the composition of various tools to accommodate a wide range of problems. Chameleon is comprised of a *module inventory* that defines different types of tools and an LLM-based *planner*, whose purpose is to decompose the original problem into sub-tasks that can be effectively solved by task-specific tools. Unlike existing tool-augmented LLM approaches [SDD23, GK23, WYQ23, SST23], our module inventory features multiple tool types as illustrated in Table 7.2, enabling Chameleon to exhibit various reasoning abilities, including image understanding, knowledge retrieval, web search, complex mathematical reasoning, and table understanding. Instead of generating domain-specific programs [NHB21, GK23, SMV23], Chameleon employs an LLM-based planner to create natural-language-like programs that follow natural language instructions, which is less error-prone, easily expandable to new modules, and user-friendly.

We formalize our planner as follows: given the input query x_0, the module inventory $\mathcal{M}$, and constraints $\mathcal{G}$, the natural language planner $\mathcal{P}$ selects a set of modules that can be executed sequentially to answer the query via generating a program in a natural-language-like format. The module inventory $\mathcal{M}$ consists of a set of pre-built modules: $\{M_i\}$, each corresponding to a tool of various types (Table 7.2). $\mathcal{G}$ are the constraints for the plan generation, for example, the concurrent relations and sequence orders of modules. In our work, the planner $\mathcal{P}$ is an LLM prompted to generate a sequence of module names in a few-shot setup. The planner is prompted in natural language with a planning task instruction $\mathcal{I}$, the descriptions of modules in $\mathcal{M}$ with corresponding constraints $\mathcal{G}$, as well as a few demonstration examples $\mathcal{D}$. A T-length plan sampled from $\mathcal{P}$ can be denoted as $p = M^1, \ldots, M^T$, where M^t represents an the t-th element in the generated plan and $M^t \in \mathcal{M}$. Formally, given an input query (problem statement) x_0, a plan p is generated as follows:

$$p \leftarrow \mathcal{P}(x_0; \mathcal{I}, \mathcal{M}, \mathcal{G}, \mathcal{D}). \tag{7.1}$$

Given the generated plan, the corresponding modules for each step are then executed sequentially. The plan is a natural-language program where each module is bound simply via string matching. When evaluating the module M^t at time step t, the output of the execution y^t is calculated by:

$$y^t \leftarrow M^t(x^{t-1}; c^{t-1}), \tag{7.2}$$

where x^{t-1} is the input for the current module M^t, and c^{t-1} is the cached information (e.g., image semantics, retrieved knowledge, generated programs) resulting from the execution history of modules. Both the problem input x^t and cache c^t for the next module M^{t+1} are updated, respectively, by:

$$x^t \leftarrow \texttt{update\_input}(x^{t-1}, y^t), \tag{7.3}$$

$$c^t \leftarrow \texttt{update\_cache}(c^{t-1}, y^t). \tag{7.4}$$

The `update_input` and `update_cache` functions are hand-designed for each M_i. Specifically, `update_input` is applied to elements in the input query, including the question, table context, and image. These elements are updated after module execution. `update_cache` corresponds to the generation of new information, such as a description for the input image or retrieved knowledge from external resources. Finally, the response r to the query is generated by the last module M^T:

$$r = y^T \leftarrow M^T(x^{T-1}; c^{T-1}). \tag{7.5}$$

Tool Types	Tools
OpenAI	Knowledge Retrieval, Query Generator, Row Lookup, Column Lookup, Table Verbalizer, Program Generator, Solution Generator
Hugging Face	Image Captioner
Github	Text Detector
Web Search	Bing Search
Python	Program Verifier, Program Executor
Rule-based	Answer Generator

Table 7.2: Different tools in our module inventory.

7.4 Applications of Chameleon

We demonstrate the applications of Chameleon on two challenging tasks: ScienceQA [LMX22] (Section 7.4.2) and TabMWP [LQC23] (Section 7.4.3), using the module inventory introduced in Section 7.4.1.

7.4.1 Module Inventory

To accommodate various reasoning capabilities over a diverse range of queries, our system utilizes a rich module inventory of various external tools. We provide a high-level overview of this inventory here, with detailed implementations in specific experiments. The complete module inventory, $\mathcal{M}$, is presented in Table 7.2. Each tool within the inventory is defined as follows:

Knowledge Retrieval: This module retrieves additional background knowledge crucial for tackling complex problems. It is especially beneficial for specialized domains like science and mathematics, providing context for the task. For example, if a query is about a tax form table, this module could generate knowledge about tax procedures, offering valuable context.

Bing Search: Like “Knowledge Retrieval”, the “Bing Search” module aims to provide wide-ranging task-relevant knowledge. In contrast, it excels when broader or up-to-date information from multiple sources is required. Using the search engine API, this module returns relevant search results based on the input query, which are then parsed and used by subsequent modules to gather richer context information from diverse sources, enhancing problem-solving effectiveness.

Query Generator: Since the original problem typically lacks a tailored query for retrieving task-relevant information, this module creates search engine queries based on the problem, which are then used by the "Bing Search" module. Mostly, it is a good strategy to use the "Query Generator" module before the "Bing Search". Coupled with the search engine tool, generating more targeted queries generally facilitates both the recall and precision of retrieved information.

Image Captioner: Designed to generate captions for images, this module provides crucial supplementary context for queries. It is particularly valuable when understanding an image semantically, like identifying objects and interactions in a scene. Using pre-trained models, it translates visual data into language, facilitating effective comprehension and reasoning about image content.

Text Detector: This module is designed to identify text within a given image. Typically, the "Text Detector" is employed when a question requires the extraction of textual information from images containing diagrams, charts, tables, maps, or other visual elements. By effectively detecting text in various formats, this module aids in the analysis and understanding of image-based content.

Row Lookup: This module is crucial when queries involve tabular context, as locating relevant cells is often required. Large tables can distract the system, so "Row Lookup" simplifies the table by retaining only rows relevant to the query. If all rows are pertinent, it returns the original table.

Column Lookup: Like the "Row Lookup" module, "Column Lookup" addresses questions involving tabular context by focusing on relevant columns. It simplifies the table by retaining only pertinent columns, or returns the original table if all columns are relevant.

Table Verbalizer: Converting structured tables into text is likely to enhance the comprehension of tabular information by various downstream modules as shown by [MCL22] for open-domain question answering, making this module a vital part of our system. It translates tables into easily understandable descriptions for modules like "Program Generator" and "Solution Generator", particularly useful for small, domain-specific tables like stem-and-leaf plots or function tables.

Program Generator: Program-aided approaches are shown to enhance the logical and mathematical reasoning abilities of LLMs [WLJ22, GMZ23, CMW23, MFL22, IDS23, PLS23]. The "Program Generator" generates Python programs to solve queries effectively, which is particularly beneficial for queries requiring complex computations or intricate logical operations, such as “if-else” statements.

Program Verifier: Recent studies highlight the importance of verification to reduce hallucination [PGH23, MTG24]. Hence, “Program Verifier” ensures the validity and error-free nature of programs generated by “Program Generator”. It checks for syntax and logical errors, and potential execution issues, enhancing the reliability and accuracy of the solutions.

Program Executor: This module executes the program generated by “Program Generator” and produces the result, bridging the gap between program generation and final solution derivation.

Solution Generator: This module generates a detailed solution to the input query using all the cached information. Employing a chain-of-thought prompting approach [WWS22b], it ensures coherent and well-structured responses. The planner can directly employ this module instead of other functional modules if it can solve the query independently, especially for simpler ones.

Answer Generator: This task-specific module uses a rule-based approach to extract and normalize answers from the results of the “Program Executor” or “Solution Generator”. Unlike the Solution Generator” that provides detailed multi-step solutions, “Answer Generator” serves as the final module in the pipeline, providing concise and task-specific answers.

7.4.2 Science Question Answering

Science Question Answering (ScienceQA [LMX22]) is a diverse benchmark for multi-modal question answering over a range of scientific topics and contexts. As examples illustrated in Figure 7.1, answering these questions requires various tools and skills like image captioning, text detection, knowledge retrieval, online resource search, and multi-clue visual reasoning. When generating programs for using tools, we limit the search space to the relevant inventory subset (Table 7.3 in the appendix). Programs are deemed invalid and default to a “Solution Generator” and “Answer Generator” sequence if these are not the final two elements, following the chain-of-thought prompting baseline [WWS22b]. See Table 7.8 in the appendix for the constructed natural language planner prompt. The prompts for LLM-based modules like “Knowledge Retrieval”, “Query Generator”, and “Solution Generator” are shown in Tables 7.10, 7.11, and 7.12, respectively, in Appendix 7.7.

7.4.3 Tabular Mathematical Reasoning

TabMWP [LQC23] is a mathematical reasoning task involving diverse tabular contexts like schedules, prices, tax forms, plots, and function relations (Figure 7.8). It requires AI systems to understand various table formats and perform precise numerical or symbolic computations. Like ScienceQA, we constrain the program search space to focus on two tool types: 1) those helping LLMs better digest tabular information (e.g., "Row Lookup", "Column Lookup", and "Table Verbalizer") and 2) those performing faithful symbolic computations (e.g., "Program Generator", "Program Verifier", and "Program Executor") as listed in Table 7.3. The generated programs must meet certain constraints, such as including "Answer Generator" and placing "Program Generator" prior to both "Program Verifier" and "Program Executor". Non-compliant programs default to a sequence of "Program Generator", "Program Verifier", "Program Executor", and "Answer Generator", aligning with the program-of-thought prompting baseline [CMW23] with added verification.

7.5 Experiments

The effectiveness and adaptability of Chameleon are evaluated on two complex reasoning tasks, ScienceQA [LMX22] and TabMWP [LQC23].

7.5.1 Experimental Details

Tool Types	Tools used on ScienceQA	Tools used on TabMWP
OpenAI	**Knowledge Retrieval**, Query Generator, **Solution Generator**	**Knowledge Retrieval**, Row Lookup, Column Lookup, Table Verbalizer, Program Generator, **Solution Generator**
Hugging Face	Image Captioner	
Github	Text Detector	
Web Search	Bing Search	
Python		Program Verifier, Program Executor
Rule-based	**Answer Generator**	**Answer Generator**

Table 7.3: Tools used on ScienceQA and TabMWP. Reusable tools are marked in green.

Planner implementations. We use the `gpt-3.5-turbo` engine for ChatGPT and the `gpt-4` engine for GPT-4 when constructing the LLM-based planner. The maximum length for generated programs is set to 128, and the temperature is set to 0 for the most deterministic generation. The planner prompts for the ScienceQA and TabMWP are illustrated in Table 7.8 and Table 7.9, respectively.

Module implementations for ScienceQA. By default, the LLM-based models use four in-context examples as demonstrations, have a temperature setting of 0, and allow a maximum of 512 tokens for completion. Additional specific implementation details are provided as follows:

- **Knowledge Retrieval**: The prompt consists of 3 demonstration examples and the template is shown in Table 7.10.
- **Query Generator**: The prompt template is shown in Table 7.11. The maximum number of tokens for completion is set as 64.
- **Solution Generator**: The prompt consists of 2 demonstration examples and the template is shown in Table 7.12.
- **Image Captioner**: We use the captioning model[1] to generate textual descriptions for input images. The maximum length of generated captions is set to 16, the number of beams is 4, and the maximum number of output tokens is 512.
- **Text Detector**: This module is based on the github model[2] to extract the text contents with coordinates in the image.
- **Bing Search**: This module calls the Bing Search API[3] and returns the top three responses for the text query.
- **Answer Generator**: This module extracts the answer snippet from the result provided by the "Solution Generator" and selects the most similar option from the given choices.

Module implementations for TabMWP. Similar to ScienceQA, the LLM-based modules by default use four in-context examples as demonstrations, have a temperature setting of 0, and allow

---

[1] `https://huggingface.co/nlpconnect/vit-gpt2-image-captioning`

[2] `https://github.com/JaidedAI/EasyOCR`

[3] `https://www.microsoft.com/bing`

a maximum of 512 tokens for completion. Additional implementation details are provided as follows:

- **Knowledge Retrieval**: The prompt consists of 5 demonstration examples and the template is shown in Table 7.13.
- **Row Lookup**: It is enabled only when there are more than three rows and 18 table cells, in order to accelerate inference. The prompt consists of 7 demonstration examples and the template is shown in Table 7.14. The maximum number of tokens for completion is set as 256.
- **Column Lookup**: Similarly, this module is enabled with two or more columns and 18 or more table cells. The prompt consists of 6 demonstration examples and the template is shown in Table 7.15. The maximum number of tokens for completion is set as 256.
- **Table Verbalizer**: The prompt consists of 7 demonstration examples and the template is shown in Table 7.16.
- **Program Generator**: The prompt template is shown in Table 7.17. The maximum number of tokens for completion is set as 256.
- **Solution Generator**: The prompt consists of 16 demonstration examples and the template is shown in Table 7.18.
- **Answer Generator**: It is used to normalize answers with two-place precision for questions with numerical answers and select the most similar option for multiple-choice questions.

Implementations of `update_input` and `update_cache`. `update_input` is triggered by the execution of specific tools, like 'Row_Lookup', which alter or replace elements in the input to reflect the updated state. Tools such as 'Image Captioner', 'Text Detector', 'Knowledge Retrieval', 'Web Search', and 'Program Generation' generate new elements. `update_cache` stores these new elements in the cache, making them accessible for later tools' execution.

7.5.2 Experimental Results

ScienceQA. Table 7.4 presents the results of existing baselines and our approach Chameleon, with key results highlighted in Figure 7.2 (a). We report the number of tuned parameters for this task and the overall accuracy, along with accuracy scores for different question types, including

Model	**#Tuned Params**	**ALL**	**NAT**	**SOC**	**LAN**	**TXT**	**IMG**	**NO**	**G1-6**	**G7-12**
Heuristic baselines										
Random Choice [LMX22]	-	39.83	40.28	46.13	29.25	47.45	40.08	33.66	39.35	40.67
Human [LMX22]	-	88.40	90.23	84.97	87.48	89.60	87.50	88.10	91.59	82.42
Fine-tuned models										
MCAN [YYC19]	95M	54.54	56.08	46.23	58.09	59.43	51.17	55.40	51.65	59.72
Top-Down [AHB18]	70M	59.02	59.50	54.33	61.82	62.90	54.88	59.79	57.27	62.16
BAN [KJZ18]	112M	59.37	60.88	46.57	66.64	62.61	52.60	65.51	56.83	63.94
DFAF [GJY19]	74M	60.72	64.03	48.82	63.55	65.88	54.49	64.11	57.12	67.17
ViLT [KSK21]	113M	61.14	60.48	63.89	60.27	63.20	61.38	57.00	60.72	61.90
Patch-TRM [LQC21]	90M	61.42	65.19	46.79	65.55	66.96	55.28	64.95	58.04	67.50
VisualBERT [LYY19, LYY20]	111M	61.87	59.33	69.18	61.18	62.71	62.17	58.54	62.96	59.92
UnifiedQA [KMK20]	223M	70.12	68.16	69.18	74.91	63.78	61.38	77.84	72.98	65.00
UnifiedQA CoT [LMX22]	223M	74.11	71.00	76.04	78.91	66.42	66.53	81.81	77.06	68.82
MM-COT$_T$ [ZZL23]	223M	70.53	71.09	70.75	69.18	71.16	65.84	71.57	71.00	69.68
MM-COT [ZZL23]	223M	84.91	87.52	77.17	85.82	87.88	82.90	86.83	84.65	85.37
MM-COT$_{Large}$ [ZZL23]	738M	91.68	95.91	82.00	90.82	95.26	88.80	92.89	92.44	90.31
LLaMA-Adapter$_T$ [ZHZ24]	1.2M	78.31	79.00	73.79	80.55	78.30	70.35	83.14	79.77	75.68
LLaMA-Adapter [ZHZ24]	1.8M	85.19	84.37	88.30	84.36	83.72	80.32	86.90	85.83	84.05
Few-shot GPT-3										
GPT-3 [BMR20]	0M	74.04	75.04	66.59	78.00	74.24	65.74	79.58	76.36	69.87
GPT-3 CoT [LMX22]	0M	75.17	75.44	70.87	78.09	74.68	67.43	79.93	78.23	69.68
Published results (Above) ▲										
Few-shot ChatGPT										
ChatGPT CoT	0M	78.31	78.82	70.98	83.18	77.37	67.92	86.13	80.72	74.03
Chameleon (ChatGPT)	0M	79.93	81.62	70.64	84.00	79.77	70.80	86.62	81.86	76.53
Few-shot GPT-4										
GPT-4 CoT	0M	83.99	85.48	72.44	90.27	82.65	71.49	92.89	86.66	79.04
Chameleon (GPT-4)	0M	**86.54**	**89.83**	**74.13**	**89.82**	**88.27**	**77.64**	**92.13**	**88.03**	**83.72**

Table 7.4: QA accuracy (%) on the test set of ScienceQA.

natural, social, and language sciences, text, image, and no context, as well as grades 1-6 and 7-12. The highest scores among models in each section and overall are highlighted in blue and red, respectively, and the results of our best model are marked in bold.

Employing ChatGPT [Ope22] as the base LLM, Chameleon achieves a 79.93% accuracy, a 1.62% improvement over Chain-of-Thought (CoT) [WWS22b] prompted ChatGPT. Notably, Chameleon is a generalized form of CoT, where the generated program is a sequence of "Solution Generator" and "Answer Generator". Chameleon benefits from additional tool usage, such as "Knowledge Retrieval", "Bing Search", "Image Captioner", and "Text Detector". When built upon GPT-4 [Ope23a], our model attains an accuracy of 86.54%, outperforming GPT-4 CoT [LMX22]

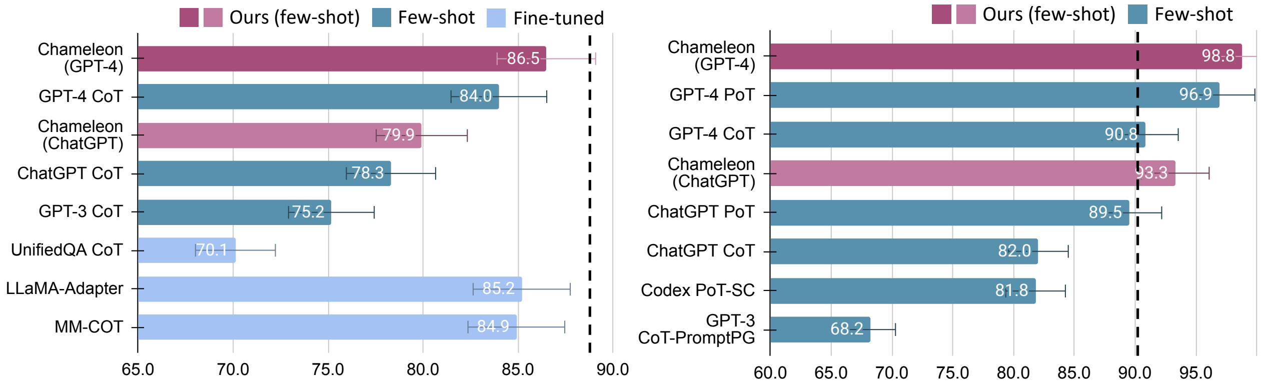
Model	**#Tuned Params**	**ALL**	**FREE**	**MC**	**INT**	**DEC**	**EXTR**	**BOOL**	**OTH**	**G1-6**	**G7-8**
Heuristic baselines											
Heuristic guess	-	15.29	6.71	39.81	8.37	0.26	30.80	51.22	26.67	17.55	12.27
Human performance	-	90.22	84.61	93.32	84.95	83.29	97.18	88.69	96.20	94.27	81.28
Fine-tuned models											
UnifiedQA$_{\text{SMALL}}$ [KMK20]	41M	29.79	22.27	51.31	27.27	2.83	52.28	48.11	69.52	35.85	21.71
UnifiedQA$_{\text{BASE}}$ [KMK20]	223M	43.52	34.02	70.68	40.74	7.90	84.09	55.67	73.33	53.31	30.46
UnifiedQA$_{\text{LARGE}}$ [KMK20]	738M	57.35	48.67	82.18	55.97	20.26	94.63	68.89	79.05	65.92	45.92
TAPEX$_{\text{BASE}}$ [LCG22]	139M	48.27	39.59	73.09	46.85	11.33	84.19	61.33	69.52	56.70	37.02
TAPEX$_{\text{LARGE}}$ [LCG22]	406M	58.52	51.00	80.02	59.92	16.31	95.34	64.00	73.33	67.11	47.07
Zero-shot GPT-3											
GPT-3 [BMR20]	0M	56.96	53.57	66.67	55.55	45.84	78.22	55.44	54.29	63.37	48.41
GPT-3 CoT [WWS22b]	0M	57.61	54.36	66.92	55.82	48.67	78.82	55.67	51.43	63.62	49.59
Few-shot GPT-3											
GPT-3 [BMR20]	0M	57.13	54.69	64.11	58.36	40.40	75.95	52.41	53.02	63.10	49.16
GPT-3 CoT [WWS22b]	0M	62.92	60.76	69.09	60.04	63.58	76.49	61.19	67.30	68.62	55.31
GPT-3 CoT-PromptPG [LQC23]	0M	68.23	66.17	74.11	64.12	74.16	76.19	72.81	65.71	71.20	64.27
Codex* [CTJ21]	0M	59.4	-	-	-	-	-	-	-	-	-
Codex PoT* [CMW23]	0M	73.2	-	-	-	-	-	-	-	-	-
Codex PoT-SC* [CMW23]	0M	81.8	-	-	-	-	-	-	-	-	-
Published results (Above) ▲											
Few-shot ChatGPT											
ChatGPT CoT	0M	82.03	78.43	92.32	75.38	90.30	92.30	92.89	87.62	83.06	80.66
ChatGPT PoT	0M	89.49	90.24	87.35	89.31	93.82	92.10	85.89	55.24	90.60	88.00
Chameleon (ChatGPT)	0M	93.28	93.13	93.72	92.71	94.76	91.29	98.11	78.85	93.37	93.17
Few-shot GPT-4											
GPT-4 CoT	0M	90.81	88.48	97.49	86.16	97.51	96.86	99.11	89.52	92.40	88.70
GPT-4 PoT	0M	96.93	97.40	95.58	98.48	93.22	96.25	98.00	68.57	96.97	96.87
Chameleon (GPT-4)	0M	**98.78**	**98.95**	**98.29**	**99.34**	**97.42**	**98.58**	**98.56**	**93.33**	**98.95**	**98.54**

Table 7.5: QA accuracy (%) on the test set of TabMWP.

by 2.55% and GPT-3 CoT by 11.37%, creating the new state of the art in few-shot settings.

TabMWP. Table 7.5 presents results with key models in Figure 7.2 (b). We report the number of tuned parameters for this task and the overall accuracy, and accuracy of different question types, including free-text questions, multi-choice questions, integer answers, decimal answers, extractive answers, Boolean answers, other text answers, grades 1-6, and grades 7-8. * refers to a subset of results.

Similarly, significant improvements are observed for Chameleon over both fine-tuned and few-shot models. It is worth noting that both CoT and Program-of-Thought (PoT) [CMW23] can be viewed as special cases of Chameleon. Apart from “Solution Generator” and “Answer Generator”,



(a) Results on ScienceQA

(b) Results on TabMWP

Figure 7.2: Results of main baselines and Chameleon. Dashed lines represent human performance.

CoT doesn't utilize any tool, while PoT only relies on symbolic programming tools like "Program Generator" and "Program Executor". Chameleon (ChatGPT) outperforms ChatGPT CoT and ChatGPT PoT by 11.25% and 3.79%, respectively, emphasizing the advantage of our enriched tool set. With GPT-4, Chameleon gains an additional 5.50%, reaching a 98.78% accuracy. Notably, Chameleon (GPT-4) surpasses Codex PoT-SC [CMW23], the best-published model, by 17.0% and human performance by 8.56%.

7.5.3 Qualitative Analysis

Tool use planning. The proportions of key tools called in the programs from Chameleon on ScienceQA and TabMWP are visualized in Figure 7.3 and Figure 7.4, respectively. Interestingly, ChatGPT and GPT-4 exhibit different planning behaviors. Generally, ChatGPT has a strong bias toward using or not using certain tools, highly influenced by in-context examples. For instance, ChatGPT calls "Knowledge Retrieval" in 72% of queries but only calls "Bing Search" in 3% of cases on ScienceQA; on TabMWP, ChatGPT heavily relies on "Row Lookup" (47%) but calls "Column Lookup" less frequently (4%). However, GPT-4 acts more *objectively* and *rationally* in tool selection. For example, GPT-4 calls "Knowledge Retrieval" more frequently (81% vs. 72%) and calls "Bing Search" more than ChatGPT (11% vs. 3%) when answering scientific questions on ScienceQA. Impressively, GPT-4 consistently calls "Query Generator" and "Bing Search" simultaneously by observing the tool usage descriptions, while ChatGPT lacks such reasoning capability.

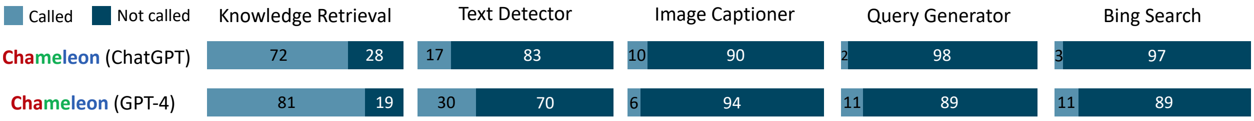


Figure 7.3: Distribution of tools called in the programs generated by Chameleon on ScienceQA.

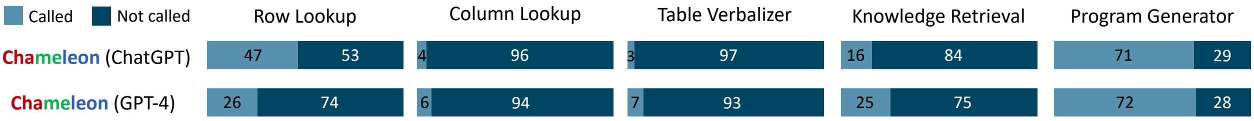


Figure 7.4: Distribution of tools called in the programs generated by Chameleon on TabMWP.

Module	**Δ (ScienceQA)**	**Δ (TabMWP)**
Knowledge Retrieval	-7.8%	-2.2%
Bing Search	-7.4%	-
Text Detector	-8.4%	-
Image Captioner	-6.0%	-
Program Generator	-	-7.4%
Table Verbalizer	-	-0.2%

Table 7.6: Score drop with disabled modules on ScienceQA and TabMWP.

Ablation study with disabled modules. We study the accuracy decline of Chameleon when key modules in the generated programs are disabled (Table 7.6), using ChaptGPT as the underlying LLMs and 500 test examples. The results reveal that "Knowledge Retrieval" plays a vital role in both tasks. Domain-specific tools, such as the search engine and vision models for ScienceQA, and program tools for TabMWP, also prove to be important.

Module transitions. We visualize the transition graphs of modules for generated programs by Chameleon (GPT-4) on ScienceQA and TabMWP in Figures 7.5 and 7.6, respectively. The transition probabilities in these graphs are computed from the tool transitions observed on the test sets. These graphs show that the GPT-4 planner is able to make good decisions on how to sequence tools in a few-shot setup. For example, on ScienceQA, Chameleon often decides to rely on either "Knowledge Retriever" or "Bing Search", but rarely both. On TabMWP, we observe two main modes: either going through the solution generator module or via the program generator, verifier, and executor.

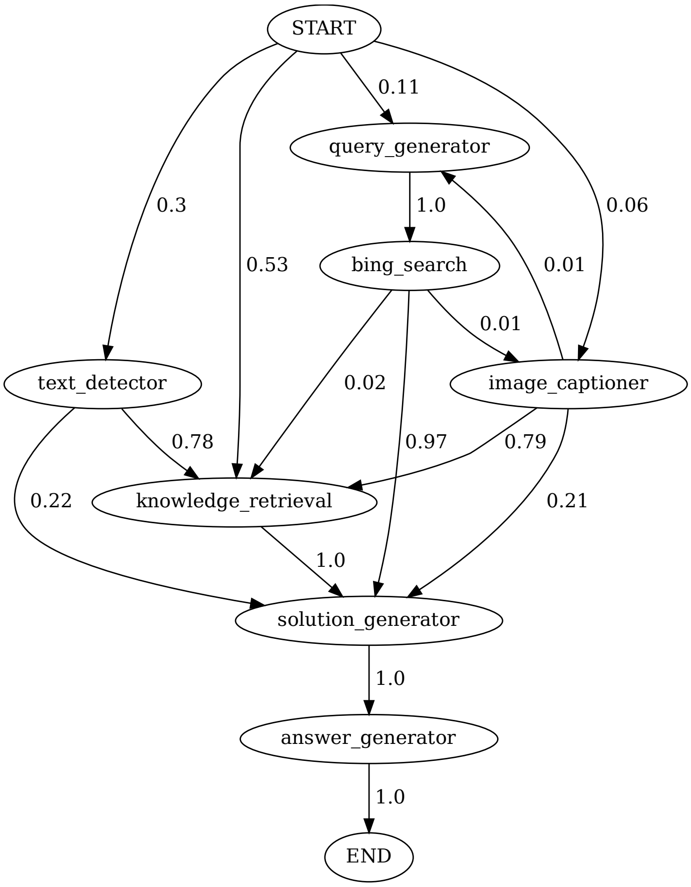


Figure 7.5: Transition probabilities between modules in programs generated by Chameleon (GPT-4) on ScienceQA. START is the start symbol, END is a terminal symbol, and the others are non-terminal symbols.

Generated program statistics. Chameleon utilizes the LLM-based natural language planner to generate programs, i.e., sequences of used modules (tools). We report the statistics of the number of unique generated programs and the average length of corresponding tool sequences by Chameleon in Table 7.7. On both ScienceQA and TabMWP, using GPT-4 as the base LLM generates fewer distinct programs, i.e., more consistent programs, than using ChatGPT, even when given the exact same prompt in the planning model. Our results are consistent with the findings in [Ope23a], which observes that GPT-4 has a superior capability of understanding long contexts, aligning with human instructions, and performing high-level reasoning compared to other LLMs such as ChatGPT.

7.5.4 Case Study

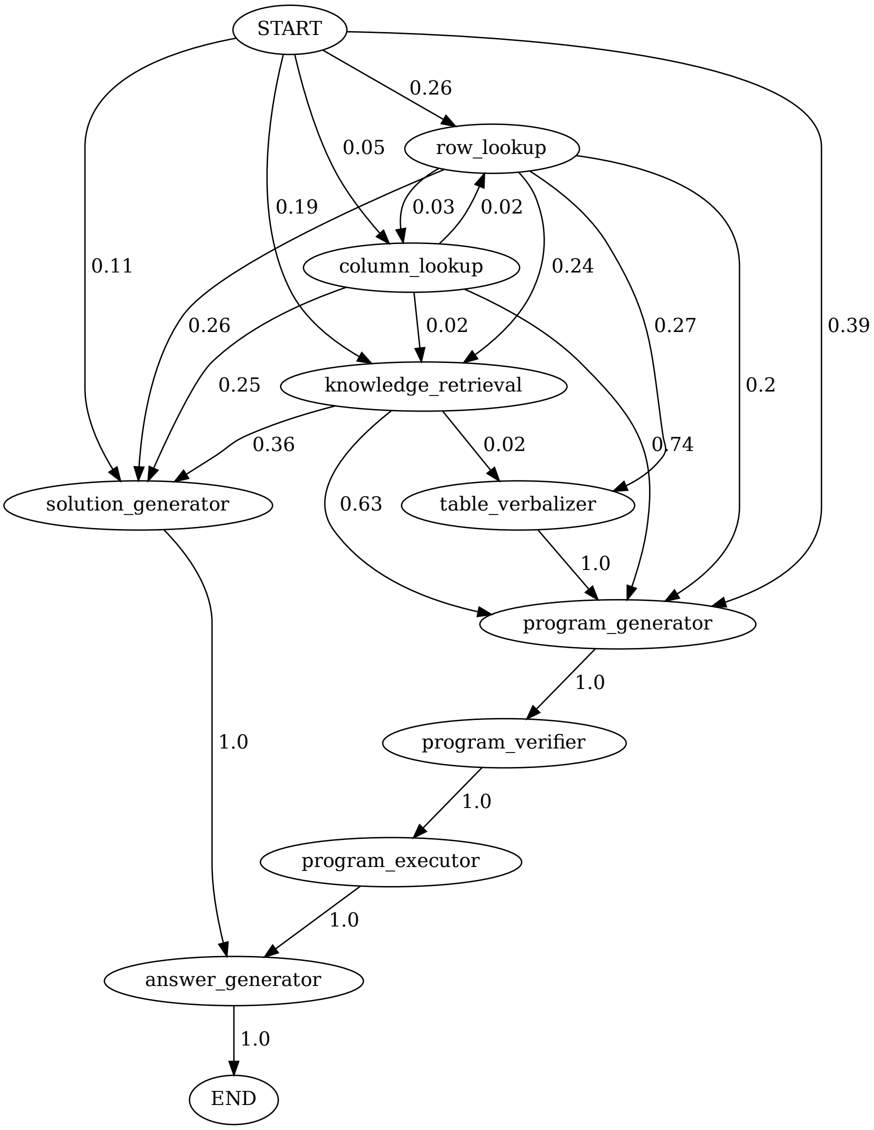
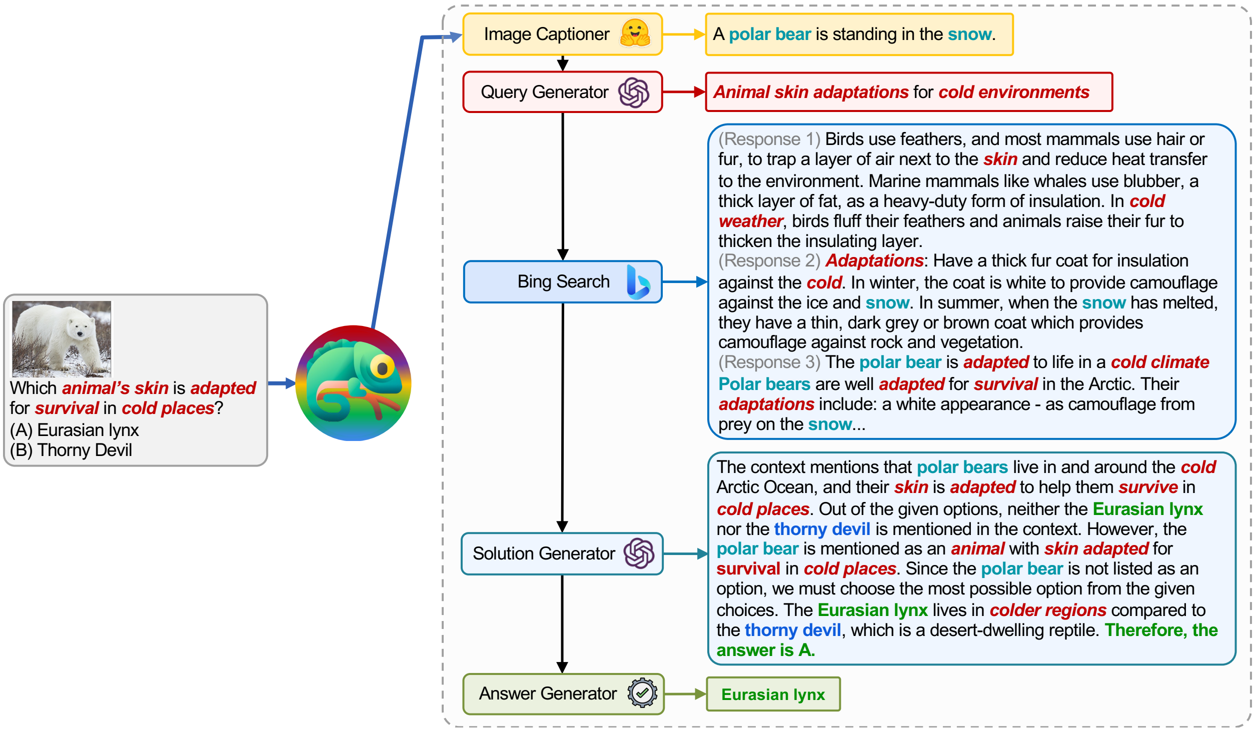


Figure 7.6: Transition probabilities between modules in programs generated by Chameleon (GPT-4) on TabMWP. START is the start symbol, END is a terminal symbol, and the others are non-terminal symbols.

Visualization examples of ScienceQA. Examples from Chameleon (GPT-4) on ScienceQA are visualized in Figure 7.1. Chameleon (GPT-4) is able to adapt to different input queries by generating programs that compose various tools and executing them sequentially to obtain accurate responses. For instance, to answer the first question (①), *What is the direction of this push?*, the system calls the image captioner model to extract semantic information from the image and employs the knowledge retrieval model to gather background knowledge for multi-modal reasoning. In the second example (②), the natural language planner infers that a text detector tool is needed

Task	Model	# of different programs	Average program length
ScienceQA	Chain-of-thought (CoT)	1	2
	Chameleon (ChatGPT)	14	3.03
	Chameleon (GPT-4)	11	3.40
TabMWP	Chain-of-thought (CoT)	1	2
	Program-of-thought (PoT)	1	3
	Chameleon (ChatGPT)	28	4.17
	Chameleon (GPT-4)	19	4.09

Table 7.7: The statistics of the number of different generated programs and the average length of generated programs by Chameleon, respectively.



Which animal's skin is adapted for survival in cold places?
(A) Eurasian lynx
(B) Thorny Devil

A polar bear is standing in the snow.

Animal skin adaptations for cold environments

(Response 1) Birds use feathers, and most mammals use hair or fur, to trap a layer of air next to the skin and reduce heat transfer to the environment. Marine mammals like whales use blubber, a thick layer of fat, as a heavy-duty form of insulation. In cold weather, birds fluff their feathers and animals raise their fur to thicken the insulating layer.
(Response 2) Adaptations: Have a thick fur coat for insulation against the cold. In winter, the coat is white to provide camouflage against the ice and snow. In summer, when the snow has melted, they have a thin, dark grey or brown coat which provides camouflage against rock and vegetation.
(Response 3) The polar bear is adapted to life in a cold climate Polar bears are well adapted for survival in the Arctic. Their adaptations include: a white appearance - as camouflage from prey on the snow...

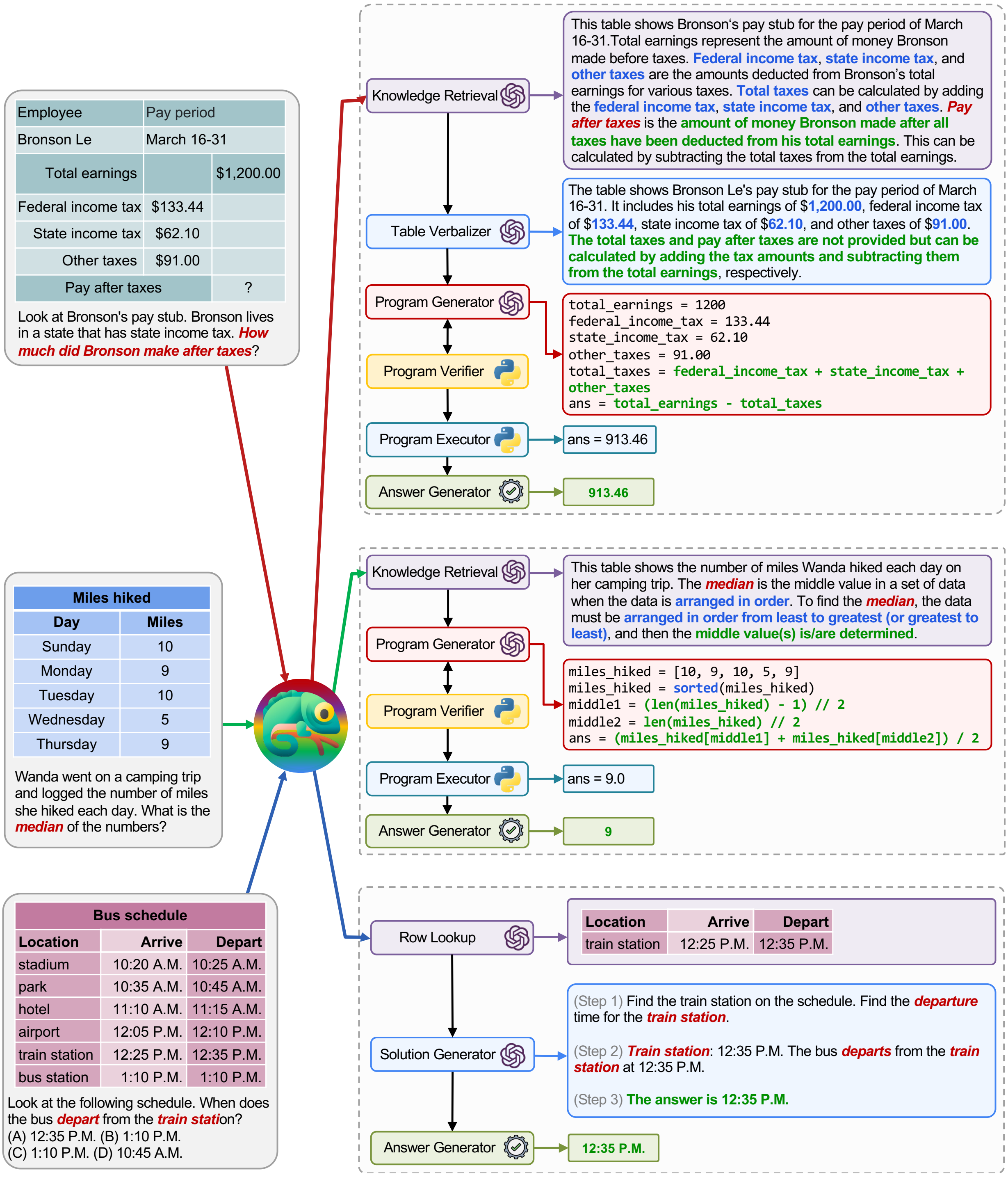
The context mentions that polar bears live in and around the cold Arctic Ocean, and their skin is adapted to help them survive in cold places. Out of the given options, neither the Eurasian lynx nor the thorny devil is mentioned in the context. However, the polar bear is mentioned as an animal with skin adapted for survival in cold places. Since the polar bear is not listed as an option, we must choose the most possible option from the given choices. The Eurasian lynx lives in colder regions compared to the thorny devil, which is a desert-dwelling reptile. Therefore, the answer is A.

Eurasian lynx

Figure 7.7: One more example from our Chameleon (GPT-4) approach on ScienceQA.

to understand the context of the ad. The third query (③; more details provided in Figure 7.7 in the appendix), *Which animal's skin is adapted for survival in cold places?*, involves scientific terminology related to animal survival. The planner decides to call the Bing search engine to access domain-specific knowledge, benefiting from the numerous online resources.

Visualization examples of TabMWP. The adaptability and versatility of Chameleon for various queries are also observed on TabMWP, as illustrated in the examples in Figure 7.8. The first



Employee	Pay period	
Bronson Le	March 16-31	
Total earnings		$1,200.00
Federal income tax	$133.44	
State income tax	$62.10	
Other taxes	$91.00	
Pay after taxes		?

Look at Bronson's pay stub. Bronson lives in a state that has state income tax. ***How much did Bronson make after taxes***?

This table shows Bronson's pay stub for the pay period of March 16-31.Total earnings represent the amount of money Bronson made before taxes. **Federal income tax**, **state income tax**, and **other taxes** are the amounts deducted from Bronson's total earnings for various taxes. **Total taxes** can be calculated by adding the **federal income tax**, **state income tax**, and **other taxes**. ***Pay after taxes*** is the **amount of money Bronson made after all taxes have been deducted from his total earnings**. This can be calculated by subtracting the total taxes from the total earnings.

The table shows Bronson Le's pay stub for the pay period of March 16-31. It includes his total earnings of $**1,200.00**, federal income tax of $**133.44**, state income tax of $**62.10**, and other taxes of $**91.00**. **The total taxes and pay after taxes are not provided but can be calculated by adding the tax amounts and subtracting them from the total earnings**, respectively.

```
total_earnings = 1200
federal_income_tax = 133.44
state_income_tax = 62.10
other_taxes = 91.00
total_taxes = federal_income_tax + state_income_tax +
other_taxes
ans = total_earnings - total_taxes
```

```
ans = 913.46
```

913.46

Miles hiked	
Day	**Miles**
Sunday	10
Monday	9
Tuesday	10
Wednesday	5
Thursday	9

Wanda went on a camping trip and logged the number of miles she hiked each day. What is the ***median*** of the numbers?

This table shows the number of miles Wanda hiked each day on her camping trip. The ***median*** is the middle value in a set of data when the data is **arranged in order**. To find the ***median***, the data must be **arranged in order from least to greatest (or greatest to least)**, and then the **middle value(s) is/are determined**.

```
miles_hiked = [10, 9, 10, 5, 9]
miles_hiked = sorted(miles_hiked)
middle1 = (len(miles_hiked) - 1) // 2
middle2 = len(miles_hiked) // 2
ans = (miles_hiked[middle1] + miles_hiked[middle2]) / 2
```

```
ans = 9.0
```

9

Bus schedule		
Location	**Arrive**	**Depart**
stadium	10:20 A.M.	10:25 A.M.
park	10:35 A.M.	10:45 A.M.
hotel	11:10 A.M.	11:15 A.M.
airport	12:05 P.M.	12:10 P.M.
train station	12:25 P.M.	12:35 P.M.
bus station	1:10 P.M.	1:10 P.M.

Look at the following schedule. When does the bus ***depart*** from the ***train station***?
(A) 12:35 P.M. (B) 1:10 P.M.
(C) 1:10 P.M. (D) 10:45 A.M.

Location	Arrive	Depart
train station	12:25 P.M.	12:35 P.M.

(Step 1) Find the train station on the schedule. Find the ***departure*** time for the ***train station***.

(Step 2) ***Train station***: 12:35 P.M. The bus ***departs*** from the ***train station*** at 12:35 P.M.

(Step 3) **The answer is 12:35 P.M.**

12:35 P.M.

Figure 7.8: Three examples of our Chameleon approach with GPT-4 on TabMWP, a mathematical reasoning benchmark with tabular contexts.

example (①) involves mathematical reasoning on a tax form. Chameleon (1) calls the knowledge retrieval model to recall basic knowledge that assists in understanding this domain-specific

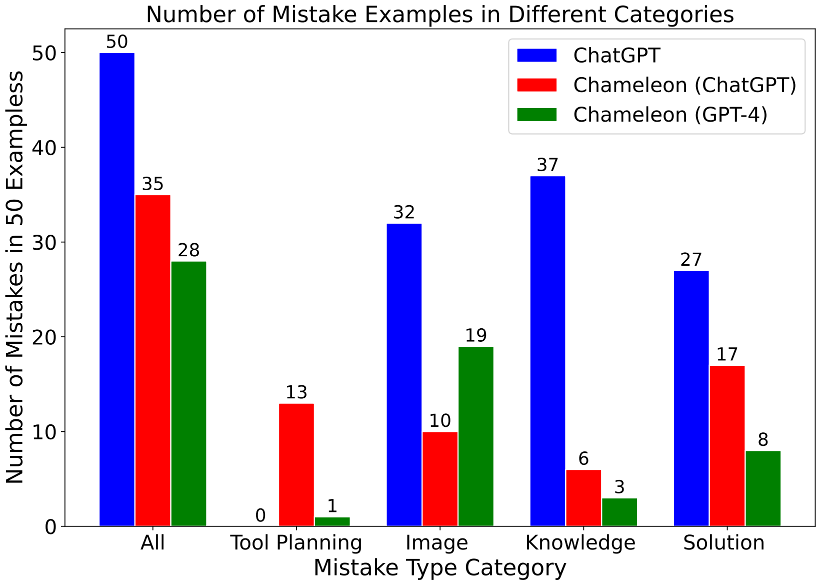


Figure 7.9: Distribution of mistake examples across different categories on ScienceQA. Image: image captioning, Knowledge: knowledge understanding, Solution: solution generation.

table, (2) describes the table in a more readable natural language format, and (3) finally relies on program-aided tools to perform precise computations. In the second example (②), the system generates Python code that closely aligns with the background knowledge provided by the knowledge retrieval model. The third example (③) requires the system to locate the cell in a large tabular context given the input query. Chameleon calls the row lookup model to help accurately locate the relevant rows and generate the language solution via an LLM model, instead of relying on program-based tools.

7.5.5 Error Analysis

To examine the error sources of the base large language models and understand how our model reduces mistakes from different aspects, we conduct an error analysis, as shown in Figure 7.9. We select 50 mistake examples from the ChatGPT baseline on ScienceQA as the evaluation set. We count the number of mistake examples and analyze their corresponding mistake type categories for ChatGPT, our Chameleon (ChatGPT) approach, and Chameleon (GPT-4).

The results show that our Chameleon approach can substantially reduce the number of mistakes compared to ChatGPT. Our model features tools for image captioning and knowledge retrieval, thus the mistakes made by ChatGPT in the category of image understanding are reduced to 10 and 19 from 32 by Chameleon (ChatGPT) and Chameleon (GPT-4); while the mistakes made by ChatGPT in the category of knowledge understanding are reduced to 6 and 3 from 37 by Chameleon (ChatGPT) and Chameleon (GPT-4). Benefiting from the sequential execution of tools, the mistakes caused by solution generation are significantly reduced as well. Additionally, we find that the task planning of GPT-4 outperforms ChatGPT by a large margin.

Failure cases. Inaccurate responses may arise from the limitations of the current modules or from suboptimal programs generated by the planner. Additionally, the module inventory may lack tools capable of addressing specific abilities. For instance, in Figure 7.19, the generated query from the "Query Generator" module is insufficiently specific, leading to a response from "Bing Search" that does not adequately detail the characteristics of the Death Valley ecosystem, particularly with regard to the types of organisms present.[4] For the example in Figure 7.20, the LLM-based solution generator struggles to understand the bus schedule, which incorporates domain-specific knowledge. Furthermore, the LLM planner does not utilize tools like "Table Verbalizer" and "Column Lookup", which could enhance the LLM's comprehension of the tabular context.

7.6 Conclusion

In conclusion, we introduce a novel *plug-and-play compositional* reasoning framework, Chameleon, that addresses the limitations of current large language models by augmenting them with external tools in a plug-and-play manner. Our approach employs a diverse set of tools and demonstrates impressive adaptability and effectiveness on two challenging benchmarks, ScienceQA and TabMWP. By achieving significant improvements in accuracy over existing state-of-the-art models, Chameleon showcases its potential for addressing real-world queries across various domains.

7.7 Appendix

---

[4] *Death Valley National Park is very much alive* from U.S. Geological Survey: `https://www.usgs.gov/geology-and-ecology-of-national-parks/ecology-death-valley-national-park-0`.

▷ *Instruction for the planner model*

You need to act as a policy model, that given a question and a modular set, determines the sequence of modules that can be executed sequentially can solve the question.

The modules are defined as follows:

Query_Generator: This module generates a search engine query for the given question. Normally, we consider using ”Query_Generator” when the question involves domain-specific knowledge.
Bing_Search: This module searches the web for relevant information to the question. Normally, we consider using ”Bing_Search” when the question involves domain-specific knowledge.
Image_Captioner: This module generates a caption for the given image. Normally, we consider using ”Image_Captioner” when the question involves the semantic understanding of the image, and the ”has_image” field in the metadata is True.
Text_Detector: This module detects the text in the given image. Normally, we consider using ”Text_Detector” when the question involves the unfolding of the text in the image, e.g., diagram, chart, table, map, etc., and the ”has_image” field in the metadata is True.
Knowledge_Retrieval: This module retrieves background knowledge as the hint for the given question. Normally, we consider using ”Knowledge_Retrieval” when the background knowledge is helpful to guide the solution.
Solution_Generator: This module generates a detailed solution to the question based on the information provided. Normally, ”Solution_Generator” will incorporate the information from ”Query_Generator”, ”Bing_Search”, ”Image_Captioner”, ”Text_Detector”, and ”Knowledge_Retrieval”.
Answer_Generator: This module extracts the final answer in a short form from the solution or execution result. This module normally is the last module in the prediction pipeline.

Below are some examples that map the problem to the modules.

▷ *In-context example(s)*

Question: Compare the average kinetic energies of the particles in each sample. Which sample has the higher temperature?

Context: The diagrams below show two pure samples of gas in identical closed, rigid containers. Each colored ball represents one gas particle. Both samples have the same number of particles.

Options: (A) neither; the samples have the same temperature (B) sample A (C) sample B

Metadata: ‘pid’: 19, ‘has_image’: True, ‘grade’: 8, ‘subject’: ‘natural science’, ‘topic’: ‘physics’, ‘category’: ‘Particle motion and energy’, ‘skill’: ‘Identify how particle motion affects temperature and pressure’

Modules: `["Text_Detector","Knowledge_Retrieval","Solution_Generator","Answer_Generator"]`

Table 7.8: The prompt constructed for the planner model on the ScienceQA task. The prompt consists of the instruction that describes the role of the planner model, the in-context examples that map the problem to the module sequence, and the test example.

▷ *Instruction for the planner model*

You need to act as a policy model, that given a question and a modular set, determines the sequence of modules that can be executed sequentially can solve the question.

The modules are defined as follows:

Program_Generator: This module generates a Python program that can solve the given question. It takes in the question and possible context and produces a program that can be executed by the ”Program_Executor” module. Normally, we consider using ”Program_Generator” when the questions and contexts involve complex computation, such as arithmetic operations over multiple numbers, or when the questions involve complex logical operations, such as ”if-else” statements.
Program_Verifier: This module verifies whether the generated program from ”Program_Generator” is valid and error-free. It checks for syntax errors, logical errors, and other potential issues that may arise during program execution.
Program_Executor: This module executes the generated program from ”Program_Generator” and produces an output that can be further processed by other modules, such as ”Question_Answering”.
Row_Lookup: This module returns the simplified table that only remains the rows that are relevant to the question. It takes in the question and a table and returns the simplified table. If all rows are relevant or there are only three rows or fewer, return the original table. Normally, we only consider using ”Row_Lookup” when the table involves more than three rows and the question only requires a small number of rows to answer the question.
Column_Lookup: This module returns the simplified table that only remains the columns that are relevant to the question. It takes in the question and a table and returns the simplified table. If all columns are relevant or there are only two columns, return the original table. Normally, we consider using ”Column_Lookup” when the table involves more than two columns and the question only requires a small number of columns to answer the question.
Table_Verbalizer: This module converts the table to a description that can be easily understood by the downstream modules, like ”Program_Generator”, ”Solution_Generator”, ”Question_Answering”. Normally, we consider using ”Table_Verbalizer” when the table involves a small number of rows and columns and the table is domain-specific, such as steam-and-leaf plots, function tables, etc.
Knowledge_Retrieval: This module retrieves domain-specific knowledge for the given question and table. Normally, we consider using ”Knowledge_Retrieval” when the question and table involve domain-specific knowledge, such as ”steam-and-leaf plots”, ”function tables”, ”tax forms”, etc.
Solution_Generator: This module generates a detailed solution to the question based on the information provided. Normally, we use ”Solution_Generator” when the question and table involve simple computation, such as arithmetic operations over a single number.
Answer_Generator: This module extracts the final answer in a short form from the solution or execution result. This module normally follows the ”Solution_Generator” or ”Problem_Executor” module.

Below are some examples that map the problem to the modules.

▷ *In-context example(s)*

Table:
designer watch — $8,141
designer coat — $6,391

Question: How much more does a designer watch cost than a designer coat? (unit: $)

Modules: `["Program_Generator","Program_Verifier","Program_Executor","Answer_Generator"]`

Table 7.9: The prompt constructed for the planner model on the TabMWP task. Similarly, the prompt consists of the instruction, the in-context examples, and the test example.

▷ *Instruction*

Read the following question, and generate the background knowledge as the context information that could be helpful for answering the question.

▷ *In-context example(s)*

Question: Which property do these three objects have in common?

Options: (A) hard (B) soft (C) yellow

Metadata: 'pid': 43, 'has_image': True, 'grade': 4, 'subject': 'natural science', 'topic': 'physics', 'category': 'Materials', 'skill': 'Compare properties of objects'

Detected text in the image: ['handkerchief', 'slippers', 'leisure suit']

Knowledge:
- This question is about comparing the properties of three objects: a handkerchief, slippers, and a leisure suit.
- The objects are related to the topic of physics and the skill of comparing properties of objects.
- Properties of objects can include physical characteristics such as color, texture, shape, size, weight, and material.

Table 7.10: The prompt constructed for the "Knowledge Retrieval" module on the ScienceQA task.

▷ *Instruction*

Read the following question and metadata, and generate the query for browser search as the context information that could be helpful for answering the question.

▷ *In-context example(s)*

Question: Which property do these two objects have in common?

Options: (A) hard (B) bendable

Metadata: 'pid': 329, 'has_image': True, 'grade': 2, 'subject': 'natural science', 'topic': 'physics', 'category': 'Materials', 'skill': 'Compare properties of objects'

Detected text in the image: [([[41, 183], [131, 183], [131, 199], [41, 199]], 'rubber gloves'), ([[245, 183], [313, 183], [313, 197], [245, 197]], 'rain boots')]

Search Query: Common material properties of jump rope and rubber gloves

Table 7.11: The prompt constructed for the "Query Generator" module on the ScienceQA task.

▷ *Instruction*

Given the question (and the context), select the answer from the options [”A”, ”B”, ”C”, ”D”, ”E”]. You should give concise and step-by-step solutions. Finally, conclude the answer in the format of ”the answer is [ANSWER]”, where [ANSWER] is one from the options [”A”, ”B”, ”C”, ”D”, ”E”]. For example, ”the answer is A”, ”the answer is B”, ”the answer is C”, ”the answer is D”, or ”the answer is E”. If the answer is not in the options, select the most possible option.

▷ *In-context example(s)*

Question: Which property do these two objects have in common?

Context: Select the better answer.

Options: (A) hard (B) bendable

Metadata: ‘pid’: 6493, ‘has_image’: True, ‘grade’: 2, ‘subject’: ‘natural science’, ‘topic’: ‘physics’, ‘category’: ‘Materials’, ‘skill’: ‘Compare properties of objects’

Image caption: A pair of scissors next to a pair of scissors.

Detected text with coordinates in the image: [([[53, 185], [121, 185], [121, 199], [53, 199]], ‘jump rope’), ([[233, 183], [323, 183], [323, 201], [233, 201]], ‘rubber gloves’)]

Retrieved knowledge:
- This question is about comparing the properties of two objects: rubber gloves and rain boots.
- The objects are related to the topic of physics and the skill of comparing properties of objects.
- Properties of objects can include physical characteristics such as color, texture, shape, size, weight, and material. In this case, the two objects have the property of being bendable in common.

Bing search response: The most common materials used for disposable gloves are Latex, Vinyl and Nitrile. Each material has its benefits and drawbacks. Latex Gloves are constructed from Natural Rubber Latex and are the most popular type of disposable glove.

Solution: An object has different properties. A property of an object can tell you how it looks, feels, tastes, or smells. Different objects can have the same properties. You can use these properties to put objects into groups. Look at each object. For each object, decide if it has that property. A bendable object can be bent without breaking. Both objects are bendable. A hard object keeps its shape when you squeeze it. The rubber gloves are not hard. The property that both objects have in common is bendable. Therefore, the answer is B.

Table 7.12: The prompt constructed for the “Solution Generator” module on the ScienceQA task.

▷ *Instruction*

Read the following table and question, and generate the domain-specific knowledge as the context information that could be helpful for answering the question.

▷ *In-context example(s)*

Table:
x | y
10 | 15
11 | 9
12 | 2

Question: The table shows a function. Is the function linear or nonlinear?

Knowledge:
- A linear function is a function whose graph is a straight line.
- A nonlinear function is a function whose graph is not a straight line.
- The equation of a linear function is $y = mx + b$, where m is the slope and b is the y-intercept.
- The equation of a nonlinear function is not $y = mx + b$.

Table 7.13: The prompt constructed for the “Knowledge Retrieval” module on the TabMWP task.

▷ *Instruction*

Read the following question and table. Each row is separated by a newline (‘\n’) and each column is separated by a vertical bar (‘|’). Return the simplified table that only remains the rows that are relevant to the question. If all rows are relevant, or the number of rows is fewer than three, return the original table.

▷ *In-context example(s)*

Question: In preparation for graduation, some teachers and students volunteered for the various graduation committees. How many people are on the music committee?

Table:
Committee | Students | Teachers
Program | 5 | 17
Ticket | 20 | 5
Music | 20 | 15
Schedule | 15 | 20
Food | 18 | 2

Simplified Table:
Committee | Students | Teachers
Music | 20 | 15

Table 7.14: The prompt constructed for the “Row Lookup” module on the TabMWP task.

▷ *Instruction*

Read the following question and table. Each row is separated by a newline ('\n') and each column is separated by a vertical bar ('|'). Return the simplified table that only remains the columns that are relevant to the question. If all columns are relevant, return the original table.

▷ *In-context example(s)*

Question: Look at the following schedule. When does Recess end?

Table:
Subject | Begin | End
Recess | 6:15 A.M. | 7:20 A.M.
Orchestra | 7:30 A.M. | 8:40 A.M.
Art | 8:45 A.M. | 9:35 A.M.
Handwriting | 9:45 A.M. | 10:20 A.M.
Gym | 10:30 A.M. | 11:15 A.M.
Choir | 11:20 A.M. | 12:25 P.M.
Science | 12:35 P.M. | 1:35 P.M.
Reading | 1:40 P.M. | 2:50 P.M.

Simplified Table:
Subject | End
Recess | 7:20 A.M.
Orchestra | 8:40 A.M.
Art | 9:35 A.M.
Handwriting | 10:20 A.M.
Gym | 11:15 A.M.
Choir | 12:25 P.M.
Science | 1:35 P.M.
Reading | 2:50 P.M.

Table 7.15: The prompt constructed for the "Column Lookup" module on the TabMWP task.

▷ *Instruction*

Read the following question and table. Write a textual description of the table. The description should keep the critical information in the table for answering the question. The description should not answer the question.

▷ *In-context example(s)*

Table:

Committee | Students | Teachers
Program | 5 | 17
Ticket | 20 | 5
Music | 20 | 15
Schedule | 15 | 20
Food | 18 | 2

Table description: The table shows the number of students and teachers on each of the four graduation committees: Program, Ticket, Music, and Schedule. The Music committee has 20 students and 15 teachers.

Table 7.16: The prompt constructed for the "Table Verbalizer" module on the TabMWP task.

▷ *Instruction*

Read the following table and then write Python code to answer a question.

▷ *In-context example(s)*

Table:

Price | Quantity demanded | Quantity supplied
$895 | 21,000 | 3,400
$945 | 17,200 | 7,400
$995 | 13,400 | 11,400
$1,045 | 9,600 | 15,400
$1,095 | 5,800 | 19,400

Questions: Look at the table. Then answer the question. At a price of $995, is there a shortage or a surplus? Please select from the following options: ['shortage', 'surplus'].

Code:

```
# Python Code, return 'ans'. Make sure that 'ans' is a string selected from
   the options in the question
quantity_demanded_at_price_955 = 13400
quantity_supplied_at_price_955 = 11400
if quantity_demanded_at_price_955 > quantity_supplied_at_price_955:
    ans = 'shortage'
else:
    ans = 'surplus'
```

Table 7.17: The prompt constructed for the "Program Generator" module on the TabMWP task.

▷ *Instruction*

Read the following table and then answer a question.

▷ *In-context example(s)*

Table:
Price | Quantity demanded | Quantity supplied
$895 | 21,000 | 3,400
$945 | 17,200 | 7,400
$995 | 13,400 | 11,400
$1,045 | 9,600 | 15,400
$1,095 | 5,800 | 19,400

Question: Look at the table. Then answer the question. At a price of $995, is there a shortage or a surplus? Please select from the following options: [‘shortage’, ‘surplus’].

Solution: At the price of $995, the quantity demanded is greater than the quantity supplied. There is not enough of the good or service for sale at that price. So, there is a shortage. The answer is shortage.

Table 7.18: The prompt constructed for the “Solution Generator” module on the TabMWP task.

▷ *Query*

Question: Which better describes the Death Valley ecosystem?

Context: Figure: Death Valley. Death Valley is a desert ecosystem in eastern California.

Options: (A) It has dry, thin soil. It also has many different types of organisms. (B) It has a small amount of rain. It also has only a few types of organisms.

Metadata: 'pid': '18973', 'has_image': True, 'grade': 3, 'subject': 'natural science', 'topic': 'biology', 'category': 'Ecosystems', 'skill': 'Describe ecosystems'

▷ *Response*

Generated Program: `["Query_Generator","Bing_Search","Solution_Qenerator","Answer_Generator"]`

Query Generator:
Characteristics of Death Valley ecosystem

Bing Search: (the first one from response candidates)
Visitors come to Death Valley to experience the stark and lonely vastness of the valley, the panorama of rugged canyons and mountains; the pleasures of the dry, moderate winter climate, the challenge of the hot, arid summer; the relief of the cooler mountains; and the reminders of frontier and Native American ways of life.

Solution Generator:
Death Valley is a desert ecosystem, which means it has specific characteristics related to its climate and the organisms that live there. A desert ecosystem typically has dry, thin soil and a small amount of rain. This environment supports only a few types of organisms that are adapted to survive in such harsh conditions. Therefore, the answer is B.

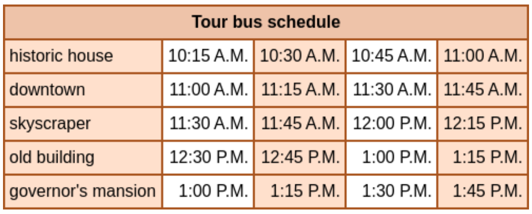
Answer Generator (Wrong prediction):
(B) It has a small amount of rain. It also has only a few types of organisms.

Correct Answer (Ground truth):
(A) It has dry, thin soil. It also has many different types of organisms.

Table 7.19: An example of failure cases in ScienceQA from Chameleon (GPT-4) where *some modules perform inaccurately*.

▷ *Query*

Table:



Tour bus schedule				
historic house	10:15 A.M.	10:30 A.M.	10:45 A.M.	11:00 A.M.
downtown	11:00 A.M.	11:15 A.M.	11:30 A.M.	11:45 A.M.
skyscraper	11:30 A.M.	11:45 A.M.	12:00 P.M.	12:15 P.M.
old building	12:30 P.M.	12:45 P.M.	1:00 P.M.	1:15 P.M.
governor's mansion	1:00 P.M.	1:15 P.M.	1:30 P.M.	1:45 P.M.

Table Title: Tour bus schedule

Table: (semi-structured format)
historic house | 10:15 A.M. | 10:30 A.M. | 10:45 A.M. | 11:00 A.M.
downtown | 11:00 A.M. | 11:15 A.M. | 11:30 A.M. | 11:45 A.M.
skyscraper | 11:30 A.M. | 11:45 A.M. | 12:00 P.M. | 12:15 P.M.
old building | 12:30 P.M. | 12:45 P.M. | 1:00 P.M. | 1:15 P.M.
governor's mansion | 1:00 P.M. | 1:15 P.M. | 1:30 P.M. | 1:45 P.M.

Question: This is Oliver's complete financial record for September. How much money did Oliver have after he purchased the basketball?

Options: (A) 2 hours and 15 minutes (B) 2 hours and 45 minutes (C) 15 minutes (D) 30 minutes

▷ *Response*

Generated Program: `["Row_Lookup","Solution_Generator","Answer_Generator"]`

Row Lookup:
historic house | 10:15 A.M. | 10:30 A.M. | 10:45 A.M. | 11:00 A.M.
governor's mansion | 1:00 P.M. | 1:15 P.M. | 1:30 P.M. | 1:45 P.M.

Solution Generator:
(Step 1) Find the last departure time from the historic house and the first arrival time at the governor's mansion:
Historic house last departure: 11:00 A.M.
Governor's mansion first arrival: 1:00 P.M.
(Step 2) Subtract the departure time from the arrival time:
1:00 P.M. - 11:00 A.M. = 2 hours
(Step 3) It takes 2 hours to get from the historic house to the governor's mansion.
(Step 4) The answer is 2 hours and 15 minutes.

Answer Generator (Wrong prediction): (A) 2 hours and 15 minutes

Correct Answer (Ground truth): (B) 2 hours and 45 minutes

Table 7.20: An example of failure cases in TabMWP from Chameleon (GPT-4) where *some modules perform inaccurately and the generated programs are suboptimal.*

Part IV

Summary and Future Directions

CHAPTER 8

Latest Advances in Visual Mathematical Reasoning

8.1 Understanding Mathematical Reasoning in Visual Contexts

A long-standing goal of AI is to create general-purpose assistants that can tackle diverse unseen tasks. We have witnessed great progress in foundation models in text such as ChatGPT [Ope22, Ope23a] and in vision-language models such as Bard [Goo23] and GPT-4V [Ope23b]. Despite these advances, a significant gap persists in our understanding of their mathematical reasoning capabilities within visual scenarios due to the lack of systematic evaluations.

Mathematical reasoning stands as a testament to the intricacies of human intelligence [Kah11]. It requires rigorous logical thinking, domain-specific knowledge, and the ability to engage in multistep reasoning processes [LKB23]. This complexity is observed not only in textual scenarios but also significantly in visual contexts. When assessing a child's mathematical reasoning capabilities, problems are often designed to encompass visual contexts [SI89, PCS20]. At the same time, AI agents with strong mathematical reasoning capabilities in visual contexts have a wide range of real-world applications, such as solving complex problems [SHF15, WLS17], addressing logical queries about statistical data [WIL23, YLW23a, LWW24], and assisting in theorem proving and scientific discovery in advanced research [TKC22, DDX23, TWL24].

Numerous datasets have been curated to assess the mathematical reasoning abilities of AI systems, with most presented purely in text form. Some datasets such as ChartQA [LGJ21, DA22, MDT22] have explored mathematical reasoning in vision-language settings. However, these datasets tend to either focus on specific tasks, like math word problems, or particular visual contexts, such as geometry problems or bar charts. General-purpose visual question answering (VQA) datasets on natural scenes contain only a small portion of questions necessitating mathematical reasoning, leaving a comprehensive investigation of vision-language reasoning within a mathematical framework largely unexplored.

On the other hand, Large Language Models (LLMs) [Ope22, Ope23a] and Vision-Language Models (VLMs) [Goo23, Ope23b, TAB23] have exhibited impressive problem-solving skills in many tasks and domains. Recently, some studies have aimed to augment existing LLMs with mathematical and scientific reasoning capabilities using external tools [LPC23, CYK24]. However, the ability of these foundation models to perform mathematical reasoning in visual contexts has not been systematically examined. Therefore, it is essential to develop a new benchmark to (1) facilitate the development of mathematical reasoning systems in visually intensive scenarios, and (2) evaluate the research progress of LLMs and VLMs, especially their capabilities in solving rigorous reasoning tasks.

We present MathVista, a consolidated **Math**ematical reasoning benchmark in **Vis**ual contexts. We propose a task taxonomy to guide the development of MathVista: (1) we identify seven mathematical reasoning types: *algebraic reasoning*, *arithmetic reasoning*, *geometry reasoning*, *logical reasoning*, *numeric common sense*, *scientific reasoning*, and *statistical reasoning*; (2) we focus on five primary tasks: *figure question answering* (FQA), *geometry problem solving* (GPS), *math word problem* (MWP), *textbook question answering* (TQA), and *visual question answering* (VQA); and (3) we encompass a diverse array of visual contexts, including natural images, geometry diagrams, abstract scenes, synthetic scenes, as well as various figures, charts, and plots.

MathVista incorporates 28 existing multimodal datasets, including 9 math-targeted question answering (MathQA) datasets and 19 VQA datasets. In addition, we have created three new datasets (*i.e.*, IQTest, FunctionQA, PaperQA) which are tailored to evaluating logical reasoning on puzzle test figures, algebraic reasoning over functional plots, and scientific reasoning with academic paper figures, respectively. Overall, MathVista consists of 6,141 examples, with 736 of them being newly curated. To facilitate fine-grained evaluation, examples are annotated with metadata, including question type, answer type, task category, grade level, visual context, and required reasoning skills.

We conduct extensive experiments in MathVista to evaluate the reasoning abilities of 12 foundation models known for their leading performance in mathematical and multimodal reasoning. This ensemble includes three LLMs (*i.e*, ChatGPT, GPT-4, Claude-2), two proprietary VLMs (*i.e.*, GPT-4V, Bard), and seven open-source VLMs. For LLMs, we examine zero-shot and few-shot settings using two prompting strategies: chain-of-thought (CoT) [WWS22b] and program-of-thought (PoT) [CMW23]. These LLMs can also be augmented with off-the-shelf visual models for image captioning and OCR. We show that MathVista, featuring advanced topics such as college curricula and scientific reasoning, is a very challenging benchmark, with human performance reaching only 60.3% accuracy.

We show that MathVista, featuring advanced topics such as college curricula and scientific reasoning, is a very challenging benchmark, with human performance reaching only 60.3% accuracy. Remarkably, GPT-4V [Ope23b], the latest multimodal version of GPT-4, achieves a state-of-the-art accuracy of 49.9%. However, a 10.4% gap in overall accuracy remains between GPT-4V and the human baseline, leaving plenty of room for model improvement. When augmented with image captions and OCR text, tool-augmented LLMs such as PoT GPT-4 are able to achieve comparable performance with VLMs like the Multimodal Bard model. We further highlight its emergent ability to drive goal-directed multi-turn human-AI dialogues.

8.2 The MathVista Benchmark

8.2.1 Collection Guidelines

Our benchmark, MathVista, is motivated to bridge the notable gap in existing benchmarks, which primarily evaluate mathematical reasoning in textual contexts. It adheres to the following collection guidelines: (1) it covers multiple tasks and topics to mirror real-world applications; (2) it incorporates diverse visual contexts and mathematical skills to foster a well-rounded evaluation; (3) it offers varying levels of challenge to effectively uncover the potential limitations of current models; and (4) it provides robust settings for deterministic evaluations.

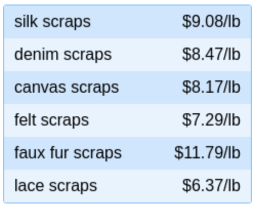
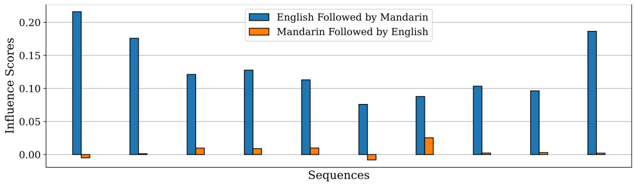
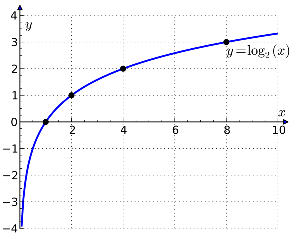
The taxonomy for this work is introduced as follows: We identify seven types of mathematical reasoning: *algebraic reasoning*, *arithmetic reasoning*, *geometry reasoning*, *logical reasoning*,

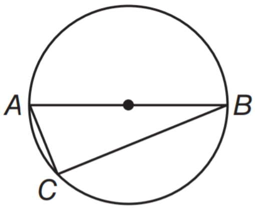
Math Reasoning	Description
Arithmetic Reasoning (34.1%)	It covers the *fundamental operations* such as addition, subtraction, multiplication, division, and understanding of n*umber properties*. It may also include the ability to interpret numerical data in different forms.
Statistical Reasoning (30.5%)	It focuses on *data interpretation* and *analysis*, including measures (mean, median, mode), dispersion metrics (standard deviation, range), probability concepts, regression, correlation, and data inferences. It also identifies trends, outliers, and patterns.
Algebraic Reasoning (28.5%)	It encompasses understanding *variables*, *equations*, and the manipulation of *expressions* with polynomials and exponents. It also covers solving simple to complex equations, and grasping functions, their properties, and graphical depictions.
Geometry Reasoning (23.3%)	It emphasizes *spatial* understanding, analysis of 2D and 3D *figures*, and reasoning about their *shapes, sizes, and relationships*. It includes symmetry, congruency, similarity, area, volume, and transformations.
Numeric Common Sense (14.0%)	It involves intuitive understanding of *daily numerical concepts*, including understanding time differences, numerical judgment, and estimates. It covers temporal reasoning, spatial numeric assessments, and practical uses like budgeting and time reading.
Scientific Reasoning (10.7%)	It deals with the application of mathematical concepts in *scientific contexts*. This includes scientific notations, formula use, understanding rates, proportions, and percentages in practical situations, and problem-solving in scientific inquiries.
Logical Reasoning (3.8%)	It focuses on *critical thinking* and *deduction* from provided information, including pattern recognition, sequence understanding, predictions, and statement evaluation. Key components include premises, conclusions, and the use of abstract reasoning.

Table 8.1: Definitions and proportions of seven mathematical reasoning categories in MathVista.

numeric common sense, *scientific reasoning*, and *statistical reasoning*, with detailed definitions provided in Table 8.1 and examples shown in Figure 8.1. We focus on five primary tasks: *figure question answering* (FQA), which centers around statistical reasoning over multiple charts and plots; *geometry problem solving* (GPS), which deals with geometrical topics; *math word problem* (MWP), which involves arithmetic reasoning in everyday scenarios; *textbook question answering* (TQA), which usually entails knowledge-intensive reasoning on scientific topics and figures; and *visual question answering* (VQA). Furthermore, our objective is to account for a diverse array of visual contexts, including natural images, geometry diagrams, abstract scenes, synthetic scenes, multiple charts and plots, scientific figures, tables, function plots, puzzle test figures, and more, with examples shown in Figure 8.2.

Therefore, We collected nine math question answer (MathQA) datasets in multimodal settings, including four for GPS, two for MWP with visual contexts of synthetic scenes, abstract diagrams, and tables, and two for TQA on college curricula. Many existing VQA datasets fea-

Math	Examples	
ARI	 silk scraps $9.08/lb denim scraps $8.47/lb canvas scraps $8.17/lb felt scraps $7.29/lb faux fur scraps $11.79/lb lace scraps $6.37/lb	**Question:** Karen bought 4 pounds of silk scraps and 4 pounds of canvas scraps. How much did she spend? (Unit: $) **Solution:** Find the cost of the silk scraps. Multiply: $9.08 × 4 = $36.32 Find the cost of the canvas scraps. Multiply: $8.17 × 4 = $32.68 Now find the total cost by adding: $36.32 + $32.68 = $69 She spent $69. **Answer:** 69
STA		**Question:** How many sequences have negative Influence Scores? **Answer:** 2
ALG		**Question:** The derivative of y at $x = 6$ is ____ that at $x = 8$. **Choices:** (A) larger than (B) equal to (C) smaller than **Answer:** (A) larger than **Question:** How many zeros does this function have? **Answer:** 1 **Question:** What is the value of y at $x = 1$? **Answer:** 0

GEO		**Question:** $\overline{AB}$ is a diameter, $AC = 8$ inches, and $BC = 15$ inches. Find the radius of the circle.
		Diagram logic forms:
		`PointLiesOnLine(D, Line(B, A))`
		`PointLiesOnCircle(B, Circle(D, radius))`
		`PointLiesOnCircle(A, Circle(D, radius))`
		`PointLiesOnCircle(C, Circle(D, radius))`
		Answer: (C) 8.5

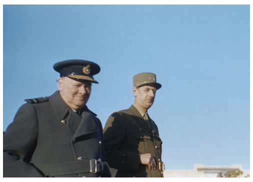
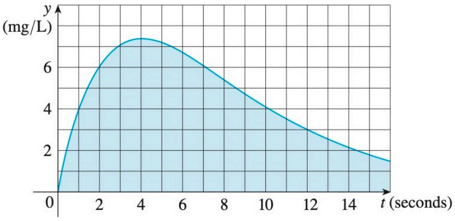
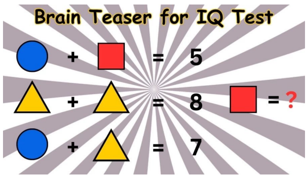
NUM		**Question:** What is the age gap between these two people in image? (unit: years) **Named entities:** Winston Churchill, Charles de Gaulle **Wiki caption**: Winston Churchill and General de Gaulle at Marrakesh, January 1944 **Answer:** 16
SCI		**Question:** The graph of the concentration function $c(t)$ is shown after a 7-mg injection of dye into a heart. Use Simpson's Rule to estimate the cardiac output. **Answer:** 5.77
LOG		**Question:** Find the value of the square in the figure. **Solution:** Circle + Square = 5, Triangle + Triangle = 8, Triangle = 4. Circle + Triangle = 7, Circle = 3. Therefore Square = 2 **Answer:** 2

Figure 8.1: Examples of seven mathematical reasoning categories in MathVista.

ture instances requiring mathematical reasoning abilities, such as arithmetic operations or numeric common sense. Therefore, reviewed more than 70 datasets, collecting 19 of them that contain

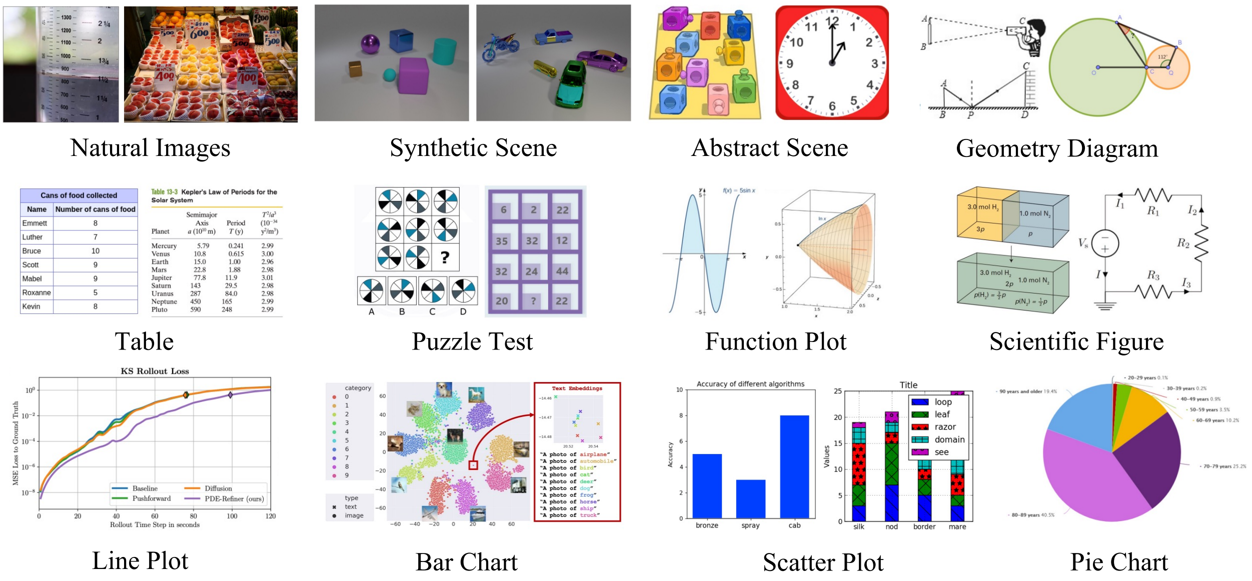
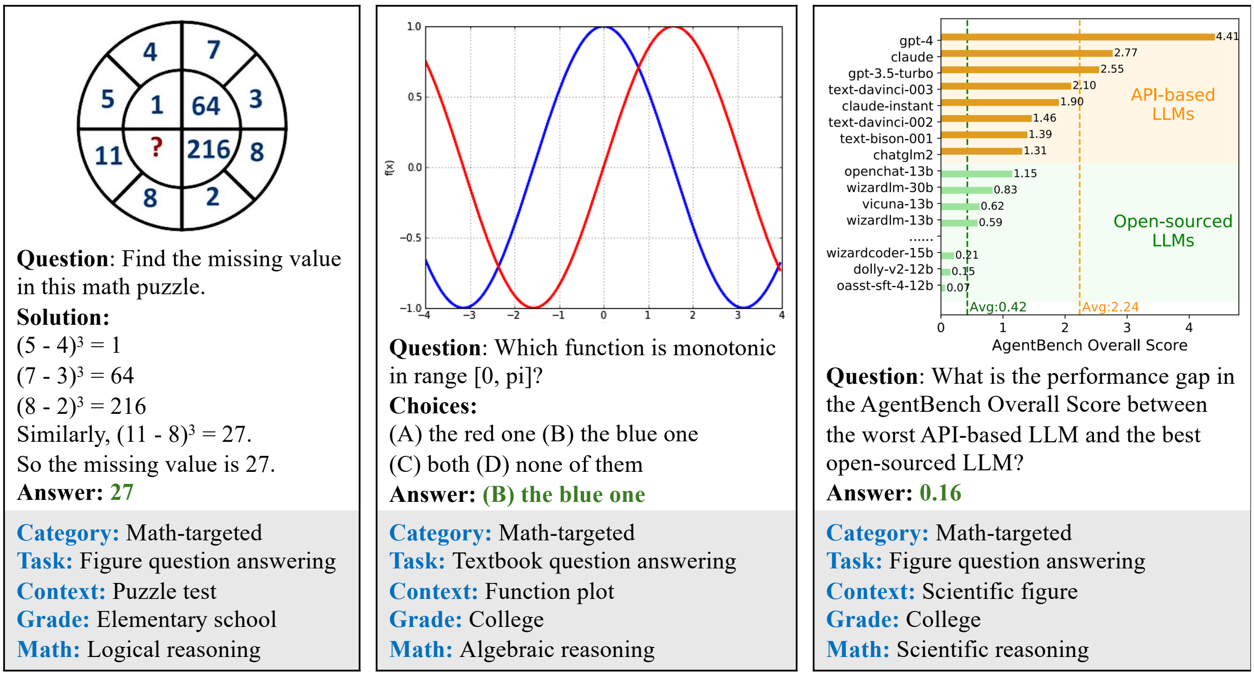


Figure 8.2: Examples of different visual contexts in MathVista.



Question: Find the missing value in this math puzzle.

Solution:

$(5 - 4)^3 = 1$

$(7 - 3)^3 = 64$

$(8 - 2)^3 = 216$

Similarly, $(11 - 8)^3 = 27$.

So the missing value is 27.

Answer: 27

Category: Math-targeted
Task: Figure question answering
Context: Puzzle test
Grade: Elementary school
Math: Logical reasoning

(a) IQTest

Question: Which function is monotonic in range [0, pi]?

Choices:

(A) the red one (B) the blue one

(C) both (D) none of them

Answer: (B) the blue one

Category: Math-targeted
Task: Textbook question answering
Context: Function plot
Grade: College
Math: Algebraic reasoning

(b) FunctionQA

Question: What is the performance gap in the AgentBench Overall Score between the worst API-based LLM and the best open-sourced LLM?

Answer: 0.16

Category: Math-targeted
Task: Figure question answering
Context: Scientific figure
Grade: College
Math: Scientific reasoning

(c) PaperQA

Figure 8.3: Examples of our newly annotated datasets: IQTest, FunctionQA, and PaperQA.

math-related instances and are publicly available.

While the source datasets we collected encompass multiple visual contexts and mathematical reasoning abilities, certain scenarios remain unaddressed: logical reasoning on puzzle test diagrams, statistical reasoning on functional plots, and scientific reasoning on academic figures. To address these gaps, we introduced three new datasets: IQTest, FunctionQA, and PaperQA, with examples illustrated in Figure 8.3. IQTest comprises examples requiring inductive reasoning, abstract thinking, pattern prediction, and calculations, sourced from puzzle test figures on online learning platforms. FunctionQA emphasizes subtle visual perceptions of functional plots and algebraic reasoning concerning variables, expressions, equations, and functions. PaperQA is a novel dataset featuring questions derived from informative academic illustrations, including tables, figures, and charts from online education resources, with examples sourced from papers released in August 2023 on Huggingface[1].

8.2.2 Benchmark Statistics

The main statistics of MathVista are presented in Table 8.2. There are two types of questions: multiple-choice and free-form. Answers to free-form questions are categorized as integers, floating numbers, or lists. The large unique number of images, questions, and answers ensures pattern diversity in MathVista. MathVista is derived from 31 source datasets, including three newly annotated datasets, IQTest, FunctionQA, and PaperQA. The distribution of the five tasks and 31 source datasets contained within MathVista is visualized in Figure 8.4. The relatively balanced distribution of these tasks enhances the benchmarking robustness that our dataset provides.

8.3 Vision-Language Models in MathVista

We aim to conduct qualitative and quantitative studies to provide a systematic evaluation of the latest foundation models for mathematical reasoning in visual contexts using MathVista.

8.3.1 Evaluation Methodologies

Recent LLMs and VLMs have been instructed to generate long responses in conventional settings instead of short text. Therefore, we propose a new strategy for benchmarking MathVista, unlike

---

[1] `https://huggingface.co/papers`

Statistic	**Number**
Total questions	6,141
- multiple-choice questions	3,392 (55.2%)
- Free-form questions	2,749 (44.8%)
- Questions with annotations	5,261 (85.6%)
- Questions newly annotated	736 (12.0%)
Unique number of images	5,487
Unique number of questions	4,746
Unique number of answers	1,464
Source datasets	31
- Existing VQA datasets	19
- Existing MathQA datasets	9
- Our newly annotated datasets	3
Maximum question length	213
Maximum answer length	27
Maximum choice number	8
Average question length	15.6
Average answer length	1.2
Average choice number	3.4

Table 8.2: Key statistics of the MathVista benchmark.

using human-designed or template matching rules [LMX22]. The evaluation process consists of three stages: *response generation*, *answer extraction*, and *score calculation*. Initially, the baselines generate responses given the input query, which incorporates the task description, the question, the choices, and the metadata. Next, the short answer text is extracted from the detailed response. We propose an answer extractor based on LLMs such as GPT-4, inspired by its remarkable ability for text processing [WWS22b]. Taking advantage of the fact that the instances in MathVista are either multiple-choice questions for textual answers or free-form questions for numerical answers, accuracy scores are used as metrics for deterministic evaluation.

8.3.2 Experimental Setup

We evaluate the models in MathVista under three setups: (a) *Text-Only LLMs* including ChatGPT [Ope22], GPT-4 [Ope23a], and Claude-2 [Ant23] in zero-shot and two-shot settings with Chain-of-Thought (CoT) [WWS22b] and Program-of-Thought (PoT) [CMW23], (b) *Augmented-LLMs* where the LLMs are provided with additional visual information including the generated image

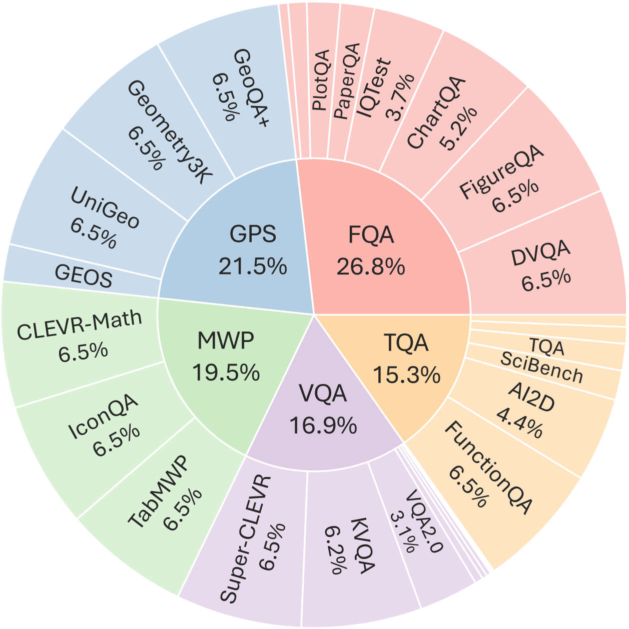


Figure 8.4: Source dataset distribution of MathVista.

captions from Multimodal Bard [Goo23] and the detected OCR text from EasyOCR [Jai20], (c) *VLMs* that include open-source models such as IDEFICS-9B [LST24], mPLUG-OWL-LLaMA-7B [YXX23], miniGPT-4-LLaMA-2-7B [ZCS23], LLaMA-Adapter-V2-7B [GHZ23], InstructBLIP-Vicuna-7B [DLL24], LLaVA-LLaMA-2-13B [LLW23], LLaVAR [ZZG23], and proprietary models such as Bard and GPT-4V. Since GPT-4V does not offer API access, we resorted to manually evaluating it using the playground chatbot.

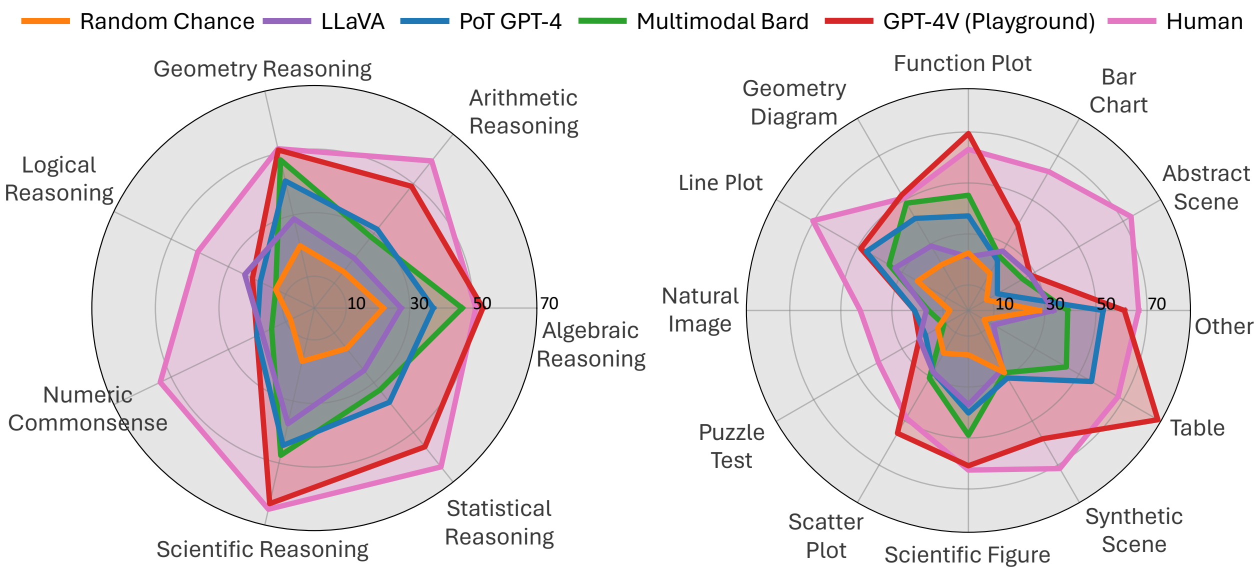
We compare the performance of several models, including Text-only LLMs, Augmented LLMs, and VLMs in MathVista in Table 8.3. We include random chance (*i.e.*, one of the options in multiple-choice questions, and empty in the free-form questions) and frequency guess as naive baselines. Additionally, we established a human performance baseline using Amazon Mechanical Turk. Eligible human annotators must have a satisfactory annotating history, successfully pass qualification examples, and possess a high school degree or higher. We asked each annotator to complete five questions within 20 minutes.

On the VLM side, Multimodal Bard scores a 34.8% accuracy, which is only 58% of human performance at 60.3%. Notably, the best-performing GPT-4V model achieves 49.9%, marking a substantial 15.1% improvement over Bard; however, it still falls 10.4% short of human performance.

Model	**Input**	**ALL**	**FQA**	**GPS**	**MWP**	**TQA**	**VQA**	**ALG**	**ARI**	**GEO**	**LOG**	**NUM**	**SCI**	**STA**
Heuristics baselines														
Random chance	-	17.9	18.2	21.6	3.8	19.6	26.3	21.7	14.7	20.1	13.5	8.3	17.2	16.3
Frequent guess	-	26.3	22.7	34.1	20.4	31.0	24.6	33.1	18.7	31.4	24.3	19.4	32.0	20.9
Large Language Models (LLMs)														
Zero-shot ChatGPT	Q only	23.5	21.9	26.9	9.1	38.6	23.5	27.7	15.9	25.7	21.6	9.9	41.5	20.5
Zero-shot GPT-4	Q only	26.1	22.3	37.0	7.0	39.2	27.4	33.6	17.4	35.6	16.2	9.2	45.8	19.5
Zero-shot Claude-2	Q only	26.4	21.9	34.1	13.4	36.1	29.1	32.8	20.4	33.3	13.5	12.1	36.4	20.5
2-shot CoT Claude-2	Q only	24.4	18.6	29.8	9.7	33.5	34.1	29.2	19.0	28.0	5.4	13.9	36.9	18.9
2-shot CoT ChatGPT	Q only	26.8	20.1	36.5	8.6	44.9	28.5	35.6	17.0	33.5	21.6	14.6	45.9	17.9
2-shot CoT GPT-4	Q only	29.2	20.1	44.7	8.6	46.2	31.3	41.6	19.3	41.0	18.9	13.9	47.5	18.9
2-shot PoT ChatGPT	Q only	25.1	19.0	30.8	16.1	38.0	25.7	29.9	19.8	29.3	24.3	19.4	38.5	16.9
2-shot PoT GPT-4	Q only	26.0	20.1	33.2	8.1	44.9	28.5	32.7	16.7	31.0	24.3	13.2	48.4	18.3
Augmented Large Language Models (Augmented-LLMs)														
2-shot CoT Claude-2	Q, I_c, I_t	33.2	26.0	31.7	35.5	48.1	30.2	32.4	32.3	33.0	16.2	17.4	54.9	36.2
2-shot CoT ChatGPT	Q, I_c, I_t	33.2	27.5	29.3	36.0	49.4	29.1	31.0	32.9	31.0	16.2	17.4	50.8	37.2
2-shot CoT GPT-4	Q, I_c, I_t	33.2	27.9	31.7	31.2	51.9	28.5	33.5	30.9	32.2	13.5	12.5	58.2	37.9
2-shot PoT ChatGPT	Q, I_c, I_t	26.8	24.5	26.4	23.7	33.5	27.9	27.8	26.1	28.0	18.9	13.2	33.6	29.9
2-shot PoT GPT-4	Q, I_c, I_t	33.9	30.1	39.4	30.6	39.9	31.3	37.4	31.7	41.0	18.9	20.1	44.3	37.9
Vision-Language Models (VLMs)														
IDEFICS-9B-Instruct	Q, I	19.8	21.6	21.1	6.5	25.9	24.0	22.1	15.0	19.8	18.9	9.9	24.6	18.1
mPLUG-Owl-LLaMA-7B	Q, I	22.2	22.7	23.6	10.2	27.2	27.9	23.6	19.2	23.9	13.5	12.7	26.3	21.4
miniGPT4-LLaMA-2-7B	Q, I	23.1	18.6	26.0	13.4	30.4	30.2	28.1	21.0	24.7	16.2	16.7	25.4	17.9
LLaMA-Adapter-V2-7B	Q, I	23.9	21.2	25.5	11.3	32.3	31.8	26.3	20.4	24.3	24.3	13.9	29.5	18.3
LLaVAR	Q, I	25.2	21.9	25.0	16.7	34.8	30.7	24.2	22.1	23.0	13.5	15.3	42.6	21.9
InstructBLIP-Vicuna-7B	Q, I	25.3	23.1	20.7	18.3	32.3	35.2	21.8	27.1	20.7	18.9	20.4	33.0	23.1
LLaVA-LLaMA-2-13B	Q, I	26.1	26.8	29.3	16.1	32.3	26.3	27.3	20.1	28.8	24.3	18.3	37.3	25.1
Multimodal Bard	Q, I	34.8	26.0	47.1	29.6	48.7	26.8	46.5	28.6	47.8	13.5	14.9	47.5	33.0
GPT-4V (Playground)	Q, I	49.9	43.1	50.5	57.5	65.2	38.0	53.0	49.0	51.0	21.6	20.1	63.1	55.8
Human														
Human performance	Q, I	60.3	59.7	48.4	73.0	63.2	55.9	50.9	59.2	51.4	40.7	53.8	64.9	63.9

Table 8.3: Accuracy scores on the *testmini* subset of MathVista. Input: Q: question, I: image, I_c: image caption, I_t: OCR text detected in the image.

These gaps highlight that there is a significant scope for further improvements on our benchmark. The open-source models (IDEFICS to LLaVA) achieve underwhelming performance in MathVista. This can be attributed to their lack of math reasoning capabilities, text recognition (useful for math word problems), shape detection (useful for geometrical problems), and chart understanding. Notably, these models utilize different model architectures for processing the vision (e.g., OpenCLIP, CLIP, Vit-G) and language (e.g., LLaMA-1, LLaMA-2), different alignment strategies (e.g., MLP projection in LLaVA, Q-former in InstructBLIP, visual abstractor in mPLUGOwl), and instruction tuning data (e.g., 150K instruction-response pairs from LLaVA data, 3,500 instruction-response pairs from miniGPT-4). While fine-tuned with instruction-following data from text-rich images, LLaVAR does not perform well, indicating that strong text recognition abilities do not guarantee high performance in MathVista, which requires comprehensive visual perception and mathematical reasoning. This underscores that there are immense possibilities for innovations in model, data, or training objectives to improve the zero-shot performance of VLMs in MathVista.



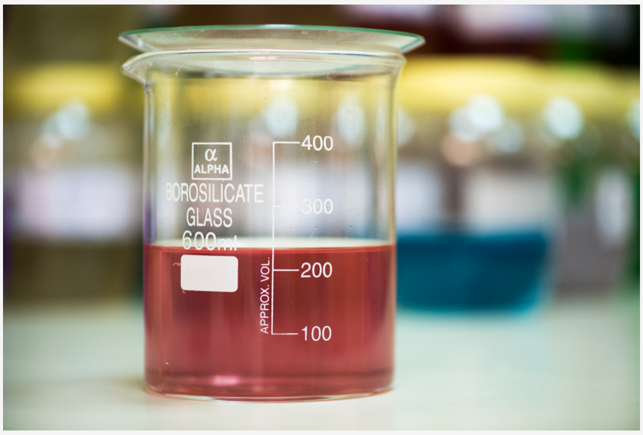
(a) Mathematical reasoning

(b) Visual context

Figure 8.5: Accuracies of one leading LLM (*i.e.*, PoT GPT-4), three prominent VLMs, random chance, and human performance on our proposed MathVista across mathematical reasoning and visual context types.

We also report fine-grained scores for a comprehensive study of the capabilities of existing models across different, mathematical reasoning abilities and visual context types in Figures 8.5. Remarkably, GPT-4V surpasses most other baselines in various categories, with exceptions in problems related to logical reasoning and numeric commonsense reasoning. Notably, GPT-4V surpasses human performance not only in tasks like geometry problem solving (GPS), textbook question answering (TQA), and mathematical reasoning skills such as algebraic reasoning but also in visual contexts including function plots, geometry diagrams, scatter plots, and tables.

We further illustrate one example in Figure 8.6. Consider the question, “What is the highest amount this glass measures?” For someone with basic scientific common sense, this beaker is able to measure 400 milliliters, rather than 600 milliliters. It appears that current advanced vision-language foundation models respond with incorrect answers. This indicates that current advanced models lack the domain-specific knowledge necessary for understanding scientific figures accurately.



ID: 19

Question: What is the highest amount this glass measures?

Ground truth: 400

LLaMA-Adapter-V2: The highest amount this glass measures is 400 ml.
Correct output: 400

LLaVA: The highest amount this glass measures is 3000 ml.

Multimodal Bard:
The highest amount the beaker in the image measures is **600 ml**.

The markings on the side of the beaker are in milliliters (ml). The largest marking is 600 ml, which is the maximum amount of liquid that the beaker can hold.

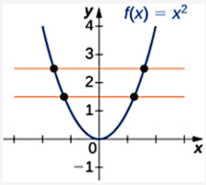
The other markings on the beaker are 300 ml, 200 ml, 100 ml, and -400 ml. The negative markings indicate the amount of liquid that is below the zero mark.

GPT-4V:
The highest amount this glass measures is 600 ml.

Figure 8.6: Most AI models without tool augmentation struggle to solve this complex mathematical reasoning example.

8.4 Tool-Augmented LLMs in MathVista

Among text-only LLMs, all models outperform the random baselines, with the 2-shot GPT-4 using chain-of-thought (CoT) prompting achieving 29.2%. The limited performance of text-only LLMs suggests that our dataset requires models to reason within visual contexts for optimal results. When equipped with image captions and detected OCR text, augmented LLMs exhibit superior performance compared to their text-only counterparts in MathVista. Specifically, the best-performing augmented LLM is the 2-shot GPT-4 employing program-of-thought (PoT) prompting, which scores 33.9%. This model generates Python programs for execution, thereby promoting rigorous reasoning.



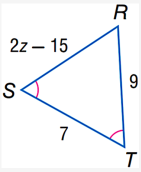
Question: Is the function (f : R to R) injective?
Choices:
(A) Yes
(B) No

Response from Bard:
The function $f(x) = x^2$ is not injective because it takes two different inputs, -1 and 1, to the same output, 1. This means that there is no one-to-one correspondence between the domain and the range of the function.

In the image, we can see that the graph of the function passes through the points $(-1, 1)$ and $(1, 1)$. This means that $f(-1) = f(1) = 1$. So, if we have two different inputs, -1 and 1, they both map to the same output, 1. This violates the definition of an injective function.

Therefore, the answer is (B).

Correct output: **(B) No**



Question: Find z.
Choices:
(A) 7 (B) 9
(C) 12 (D) 15

Response from Bard:
We can use the Law of Cosines to solve for z. The Law of Cosines states that for a triangle with side lengths a, b, and c and angle measure θ opposite side c, the following equation holds: $c^2 = a^2 + b^2 - 2ab\cos(\theta)$

In this case, we have $a = 7$, $b = 9$, and $\theta = 90^\circ$ (because $\triangle RTS$ is a right triangle). Plugging these values into the Law of Cosines, we get: $z^2 = 7^2 + 9^2 - 2(7)(9)\cos(90^\circ)$

Since $\cos(90^\circ) = 0$, we can simplify the equation to get: $z^2 = 49 + 81 = 130$

Taking the square root of both sides, we get $z = \sqrt{130}$. Since $\sqrt{130}$ is not among the answer choices, we round z to the nearest integer, which is $\boxed{12}$.

Correct output: **(C) 12**

(a) **Correct answer** and **explanation** (b) **Correct answer** but **wrong explanation**

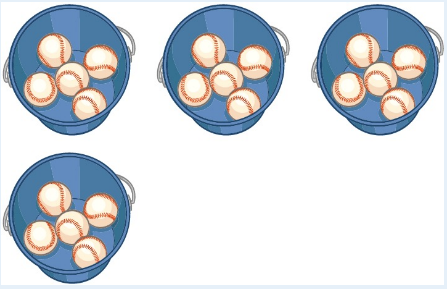
Figure 8.7: Two examples showcasing GPT-4's performance on MathVista, which depends on the quality of generated captions and detected OCR texts.

Augmented with external visual models, CoT GPT-4 and PoT GPT-4 are able to achieve comparable performance with Multimodal Bard. As shown in Figure 8.7 (a), provided with the accurate OCR text detected in the image, PoT GPT-4 accurately understands the structural information of the image and generates a code snippet to perform precise statistical reasoning. In Figure 8.7 (b), the caption provides some accurate descriptions of the image (e.g., $f(x) = c$) along with hallucination (e.g., $y = 3$, the line passes through $(0, 3)$) caused by the external Bard model. Although CoT GPT-4 predicts the correct answer given the partially correct information, the qualities of visual information augmented by external models have an impact on the accurate visual perception and thus the final mathematical reasoning performance.

8.5 Exploring Human-AI Collaborations

This section investigates the use of GPT-4V for multi-turn human-AI interaction in MathVista. We found that GPT-4V is effective in engaging multi-turn goal-directed conversations with users. In particular, GPT-4V can make good use of *hints* (*e.g.,* user feedback or responses) to guide the conversion to generate desirable results. For instance, it can rectify visual perception errors based on hints. Given the visual question “How many baseballs are there?” in Figure 8.8, GPT-4V initially produces an incorrect solution due to errors in detecting the number of baseballs. After receiving user feedback, like “Your solution is wrong,” it recalibrates and generates the accurate answer.

It can also refine reasoning steps and calculations. For instance, in Figure 8.9, there is a question “which number comes next,” given consecutive numbers 13, 57, 911, 1315, and 1719. GPT-4V correctly identifies the sequence of numbers in the image but struggles to discern the underlying pattern as it infers the next number is a concatenation of prime numbers 23 and 29. However, this question does not involve prime numbers. With feedback from the user, it re-evaluates the pattern and produces the correct response.



How many baseballs are there?

Let's count the baseballs in each bucket:

1. First bucket: 5 baseballs
2. Second bucket: 6 baseballs
3. Third bucket: 6 baseballs
4. Fourth bucket: 6 baseballs

Total baseballs: 5 + 6 + 6 + 6 = 23 baseballs.

Answer: 23.

Your solution is wrong. You counted the baseballs in each bucket incorrectly.

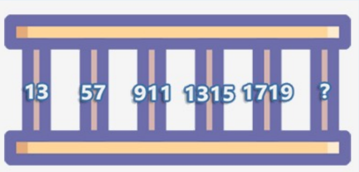
I apologize for the oversight. Let's try counting again:

1. First bucket: 5 baseballs
2. Second bucket: 5 baseballs
3. Third bucket: 5 baseballs
4. Fourth bucket: 5 baseballs

Total baseballs: 5 + 5 + 5 + 5 = 20 baseballs.

Answer: 20.

Figure 8.8: GPT-4V initially produces an incorrect solution due to visual perception errors but recalibrates after receiving user feedback.



Which number comes next?

The pattern seems to involve prime numbers.

Let's break it down:
1 and 3 are the first and second prime numbers.
5 and 7 are the third and fourth prime numbers.
9 (not a prime) but followed by 11 which is the fifth prime number.
13 is the sixth prime number and 15 (not a prime) followed by 17 which is the seventh prime number.
19 is the eighth prime number.

Following this pattern, the next two prime numbers are 23 and 29.

So, the next number in the sequence would be 2329.

Your solution is incorrect. The pattern does not involve prime numbers.

I apologize for the oversight. Let's re-examine the pattern.

It seems the pattern is derived by concatenating consecutive numbers:

1 and 3 give 13.
5 and 7 give 57.
9 and 11 give 911.
13 and 15 give 1315.
17 and 19 give 1719.

Following this pattern, the next numbers to be concatenated would be 21 and 23.

So, the next number in the sequence would be 2123.

Figure 8.9: GPT-4V correctly identifies the number sequence but struggles with the underlying pattern until receiving user feedback.

CHAPTER 9

Future Work and Conclusion

9.1 Current Challenges and Future Prospects

9.1.1 Analyzing Performance Saturation and Bridging Performance Gaps

As indicated in Table 9.1, current advanced LLMs are approaching performance saturation on mathematical reasoning benchmarks in textual contexts. For example, Claude 3 Opus [Ant24] attains a 95.0% accuracy rate on GSM8K [CKB21], a collection of grade-school mathematics questions, leaving a slim margin of 5.0% for further improvement. In the case of MATH [HBK21], a set of college-level mathematics problems, the program-aided GPT-4 model [ZWL23] achieves an accuracy of 84.3%, with a remaining gap of 15.7%. However, it is essential to acknowledge the substantial challenges AI models face in addressing mathematical reasoning within visual contexts. In the recently developed MathVista [LBX24], a benchmark for math reasoning in visual contexts, the advanced GPT-4V model [Ope23b] scores 49.9%, and InternVL-Chat-V1.2-Plus [CWW23], the state-of-the-art (SOTA) model, reaches 59.9%. Both scores significantly lag behind satisfactory performance. This discrepancy likely stems from the combined challenges of understanding and reasoning within multimodal settings, as current visual LLMs exhibit limited capabilities in comprehending fine-grained semantics presented in visual inputs and aligning different modalities for reliable predictions [LBX24].

Looking ahead, there are several promising avenues for overcoming the current limitations and enhancing the capabilities of LLMs in mathematical reasoning, especially within visual contexts. First, there is a critical need for developing more robust and challenging benchmarks. These benchmarks should not only test the current capabilities of LLMs but also encourage advancements in understanding complex, multimodal mathematical problems. Second, one of the significant

Dataset	**Model**	**Setting**	**Accuracy**	**Gap**
Textual contexts				
GSM8K	Claude 3 Opus [Ant24]	0-shot CoT	95.0%	**5.0%**
GSM8K	GPT-4 [Ope23a]	5-shot CoT	92.0%	8.0%
GSM8K	Gemini 1.5 Pro [Tea24]	11-shot CoT	91.7%	8.3%
MATH	GPT-4-Code [ZWL23]	-	84.3%	**15.7%**
MATH	Claude 3 Opus [Ant24]	Maj@32 4-shot	73.7%	26.3%
MATH	Claude 3 Opus [Ant24]	0-shot	60.1%	39.9%
Visual contexts				
MathVista (testmini)	InternVL-Chat-V1.2-Plus [CWW23]	0-shot CoT	59.9%	**40.1%**
MathVista (testmini)	Gemini 1.0 Ultra [TAB23]	0-shot CoT	53.0%	47.0%
MathVista (testmini)	GPT-4V [LBX24]	0-shot CoT	49.9%	51.1%

Table 9.1: State-of-the-art performance scores of mathematical reasoning benchmarks in textual contexts (MATH and GSM8K) and visual contexts (MathVista). Statistics date: March 5, 2024.

challenges in visual contexts is aligning textual and visual information to enable comprehensive reasoning. Future work will benefit from creating more effective methodologies for this alignment, allowing visual LLMs to better understand and integrate multimodal data. Last but not least, incorporating external tools and resources into LLM frameworks has shown promise in textual mathematical reasoning [LBX24]. Tool-augmented models in visual settings could leverage diagrammatic solvers, geometric inference engines, or other specialized tools to enhance the LLMs' ability to interpret and reason about visual mathematical problems.

9.1.2 Strategies for Self-Improvement

Supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) on labeled data, such as human-collected datasets, play a crucial role in enhancing the performance of LLMs on specialized tasks like mathematical reasoning. This process heavily depends on the availability and quality of annotated data, which often poses a bottleneck in mathematics due to the limited availability of high-quality labeled datasets. An emerging solution to this challenge involves leveraging model-generated synthetic data to expand the training datasets for LLMs [SCA23, GPS23, CDY24]. Specifically, an LLM can undergo further fine-tuning using synthetic data generated either by itself or by an external LLM.

Pioneering efforts such as ToRA [GSG23], MathCoder [WRZ23], and MathGenie [LZR24] have prompted teacher LLMs like GPT-4 [Ope23a] and Llama-2 [TMS23] to produce augmented mathematical questions and solutions. Despite these advancements, it remains relatively unexplored how mathematical LLMs can directly benefit from self-generated synthetic data, bypassing the need for external, more advanced teacher LLMs. Moreover, while initial research indicates that LLMs may show improvement when trained on synthesized data during the early iterations, there appears to be a plateau in performance gains in subsequent iterations. This suggests an inherent upper bound to self-improvement that can be achieved through synthetic data alone. Identifying and overcoming this upper bound will require innovative approaches that can stimulate continuous learning and adaptation in LLMs, potentially involving more sophisticated data generation techniques, iterative refinement processes, and mechanisms for evaluating and integrating novel insights generated by the models themselves.

9.1.3 Retrieval and Tool-augmented Algorithms

An additional focus is on innovating retrieval and tool-augmented algorithms to amplify LLMs' capabilities in rigorous reasoning and scientific discovery. Our prior work, Chameleon [LPC23], has demonstrated that retrieval and tool-augmented LLMs can enhance the reasoning performance of LLMs in mathematical and scientific reasoning within multimodal contexts. These benefits stem from access to vision models, parametric knowledge, online resources, Python interpreters, and specialized tools from promoted LLMs. This proof-of-concept idea is further verified in our subsequent work, which integrates Wolfram—a computational and scientific tool—to address college-level scientific problems. In the future, retrieval and tool-augmented algorithms for LLMs can play important roles in achieving expert-level reasoning abilities in challenging areas by incorporating more computational tools, scientific platforms, and domain aspects. For instance, this expansion will involve augmenting LLMs to seamlessly utilize domain-specific tools and integrate expert insights in biomedical research processes. Furthermore, by developing LLM agents capable of autonomously retrieving online resources and integrating external feedback, such as execution information and test case outcomes, these models could effectively address coding challenges, exemplified by those in the International Olympiad in Informatics (IOI).

9.1.4 Scientific Discovery and Autonomous Scientists

LLMs and VLMs have emerged as powerful tools for mathematical discovery, leveraging their vast knowledge base and computational abilities to solve complex problems. The potential of LLMs in this field has been significantly advanced by the introduction of FunSearch [RBN23], a novel evolutionary procedure that has made substantial strides in the realms of extremal combinatorics, particularly with the cap set problem, and in tackling algorithmic challenges such as online bin packing. In scientific domains, LLMs have emerged as promising models in generating dermatological diagnoses [LXB24], discovering statistical models [LFG24], predicting translation efficiency and mRNA expression levels [CYL24], and designing scientific experiments [BMK23, HQC24].

In the future, it might be promising to design automated systems capable of proposing hypotheses, planning experiments, analyzing outcomes, and refining hypotheses. One example in the medical domain could be the creation of LLMs and VLMs with a vast knowledge base spanning biology, chemistry, and medicine, capable of interpreting a variety of data, including academic papers, databases, and experimental results. These models will proficiently process diverse data formats, from natural language and LaTeXto amino acid and RNA sequences, enabling them to interpret complex documents, answer intricate queries, and generate innovative hypotheses.

9.2 Conclusion

In this dissertation, we have explored the frontier of mathematical reasoning within the context of language models, focusing particularly on multimodal and knowledge-intensive approaches. Through the introduction of innovative benchmarks [LGJ21, LQC21, LMX22, LQC23], we have systematically evaluated the abilities of large language models (LLMs) and vision-language models (VLMs) to process and understand complex mathematical and scientific concepts embedded in real-world scenarios.

Our work has significantly advanced the field by developing new methodologies such as Inter-GPS [LGJ21], a neuro-symbolic solver for geometry that achieves average human-level performance for the first time in history, and Chameleon [LPC23], a tool-augmented reasoning framework that enhances the reasoning capabilities of LLMs through external computational tools. The novel approach of dynamic prompting via policy gradient, referred to as PromptPG, has emerged as a pivotal development, allowing for the dynamic selection of in-context examples via reinforcement learning to enhance the few-shot learning capabilities of large language models. Additionally, we introduced the Patch-TRM model [LQC21], which integrates patch-based cross-modal transformers to enhance visual mathematical reasoning. These methodologies have not only achieved state-of-the-art performance but have also opened new avenues for research in AI, particularly in areas requiring deep multimodal understanding and the integration of external knowledge sources.

The journey of this research explored various aspects of mathematical reasoning within textual and visual contexts, revealing both the potential and limitations of current models. While substantial progress has been made in textual contexts, visual mathematical reasoning remains a formidable challenge, as evidenced by the less satisfactory performance in benchmarks like MathVista [LQY23]. This highlights a crucial area for future research, where more robust and comprehensive benchmarks could be developed to push the capabilities of language models further, especially in interpreting complex visual information.

Looking forward, the challenges that remain are vast yet invigorating. The observed performance saturation in specific models prompts a deeper exploration into innovative training paradigms and algorithm enhancements. We also foresee significant developments in self-improving systems, which autonomously refine their capabilities through continuous learning cycles. Furthermore, the integration of retrieval and tool-augmented algorithms will likely play a crucial role in enhancing model performance across complex domains. Additionally, the pursuit of autonomous scientific discovery systems presents an exciting frontier that could redefine how research is conducted.

In conclusion, the integration of AI into mathematical and scientific reasoning is just beginning. This dissertation not only contributes significant knowledge to the academic community but also opens numerous avenues for future research. The frameworks and benchmarks introduced provide a solid foundation for further investigation, pushing forward the boundaries of what AI techniques such as language models can achieve in complex reasoning tasks and facilitating scientific discoveries.

REFERENCES

[AAL15] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. "VQA: Visual question answering." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2425–2433, 2015.

[AGL19] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. "MathQA: Towards interpretable math word problem solving with operation-based formalisms." In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2019.

[AGM14] Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. "Synthesis of geometry proof problems." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[AGM17] Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. "Synthesis of solutions for shaded area geometry problems." In *The Thirtieth International Flairs Conference*, 2017.

[AHB18] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. "Bottom-up and top-down attention for image captioning and visual question answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Ant23] Anthropic. "Claude 2.", 2023.

[Ant24] Anthropic. "The Claude 3 model family: Opus, Sonnet, Haiku.", 2024.

[ARD16a] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. "Learning to compose neural networks for question answering." In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 1545–1554, 2016.

[ARD16b] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. "Neural module networks." In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39–48, 2016.

[BBV18] Mislav Balunovic, Pavol Bielik, and Martin T Vechev. "Learning to solve SMT formulas." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10338–10349, 2018.

[BCL21] Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. "FLEX: Unifying evaluation for few-shot NLP." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[BGL14] Mohit Bansal, Kevin Gimpel, and Karen Livescu. "Tailoring continuous word representations for dependency parsing." In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 809–815, 2014.

[BHB18] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. "Distributed distributional deterministic policy gradients." In *International Conference on Learning Representations (ICLR)*, 2018.

[BMK23] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. "Autonomous chemical research with large language models." *Nature*, **624**(7992):570–578, 2023.

[BMR20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1877–1901, 2020.

[BNM17] Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. "RelNet: End-to-end modeling of entities & relations." *arXiv preprint arXiv:1706.07179*, 2017.

[Bob64] Daniel G Bobrow. "Natural language input for a computer problem solving system." *AI Technical Reports*, 1964.

[CCE18] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. "Think you have solved question answering? try ARC, the AI2 reasoning challenge." *arXiv preprint arXiv:1803.05457*, 2018.

[CCS20] Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W Cohen. "Open question answering over tables and text." In *International Conference on Learning Representations (ICLR)*, 2020.

[CCS21] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. "FinQA: A dataset of numerical reasoning over financial data." In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3697–3711, 2021.

[CDY24] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. "Self-play fine-tuning converts weak language models to strong language models." In *International Conference on Machine Learning (ICML)*, 2024.

[CGZ96] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. "Automated generation of readable proofs with geometric invariants." *Journal of Automated Reasoning*, **17**(3):325–347, 1996.

[Chi98] Mohan Chinnappan. "Schemas and mental models in geometry problem solving." *Educational Studies in Mathematics*, **36**:201–217, 1998.

[CHL24] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. "Scaling instruction-finetuned language models." *Journal of Machine Learning Research*, **25**(70):1–53, 2024.

[CJL19] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. "Class-balanced loss based on effective number of samples." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9268–9277, 2019.

[CKB21] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. "Training verifiers to solve math word problems." *arXiv preprint arXiv:2110.14168*, 2021.

[CLQ22] Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. "UniGeo: Unifying geometry logical reasoning via reformulating mathematical expression." In *The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[CLY20] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. "UNITER: Universal image-text representation learning." In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[CMB14] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the properties of neural machine translation: Encoder-decoder approaches." In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, 2014.

[CMG14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

[CMW23] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks." *Transactions on Machine Learning Research*, 2023.

[CND23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. "PaLM: Scaling language modeling with pathways." *Journal of Machine Learning Research*, **24**(240):1–113, 2023.

[CTJ21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. "Evaluating large language models trained on code." *arXiv preprint arXiv:2107.03374*, 2021.

[CWG19] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. "Learning imbalanced datasets with label-distribution-aware margin loss." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1567–1578, 2019.

[CWW23] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Zhong Muyan, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. "InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks." *arXiv preprint arXiv:2312.14238*, 2023.

[CYK23] Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Elaine Wan, Xueguang Ma, Jianyu Xu, Tony Xia, and Xinyi Wang. "TheoremQA: A theorem-driven question answering dataset." In *The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[CYK24] Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Elaine Wan, Xueguang Ma, Jianyu Xu, Tony Xia, and Xinyi Wang. "SciBench: Evaluating college-level scientific problem-solving abilities of large language models." In *International Conference on Machine Learning (ICML)*, 2024.

[CYL24] Yanyi Chu, Dan Yu, Yupeng Li, Kaixuan Huang, Yue Shen, Le Cong, Jason Zhang, and Mengdi Wang. "A $5'$ UTR language model for decoding untranslated regions of mRNA and function predictions." *Nature Machine Intelligence*, pp. 1–12, 2024.

[CZC20] Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. "HybridQA: A dataset of multi-hop question answering over tabular and textual data." In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1026–1036, 2020.

[DA22] Adam Dahlgren Lindström and Savitha Sam Abraham. "CLEVR-Math: A dataset for compositional language, visual and mathematical reasoning." In *16th International Workshop on Neural-Symbolic Learning and Reasoning*, volume 3212, 2022.

[DBK21] 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." In *The International Conference on Learning Representations (ICLR)*, 2021.

[DDX23] Qingxiu Dong, Li Dong, Ke Xu, Guangyan Zhou, Yaru Hao, Zhifang Sui, and Furu Wei. "Large language model for science: A study on P vs. NP." *arXiv preprint arXiv:2309.05689*, 2023.

[DJT21] Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. "Explaining answers with entailment trees." *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[DLL24] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. "InstructBLIP: Towards general-purpose vision-language models with instruction tuning." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.

[Fei63] Edward A Feigenbaum et al. *Computers and thought.* McGraw-Hill, 1963.

[FMS17] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm." In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1615–1625, 2017.

[GGB20] Mor Geva, Ankit Gupta, and Jonathan Berant. "Injecting numerical reasoning skills into language models." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 946–958, 2020.

[GHL60] Herbert Gelernter, James R Hansen, and Donald W Loveland. "Empirical explorations of the geometry theorem machine." In *Western Joint IRE-AIEE-ACM Computer Conference*, pp. 143–149, 1960.

[GHZ23] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, Hongsheng Li, and Yu Qiao. "LLaMA-Adapter V2: Parameter-efficient visual instruction model." *arXiv preprint arXiv:2304.15010*, 2023.

[GJY19] Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven CH Hoi, Xiaogang Wang, and Hongsheng Li. "Dynamic fusion with intra-and inter-modality attention flow for visual question answering." In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6639–6648, 2019.

[GK23] 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 (CVPR)*, pp. 14953–14962, 2023.

[GKS17] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. "Making the V in VQA matter: Elevating the role of image understanding in visual question answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6904–6913, 2017.

[GLL18] Peng Gao, Hongsheng Li, Shuang Li, Pan Lu, Yikang Li, Steven CH Hoi, and Xiaogang Wang. "Question-guided hybrid convolution for visual question answering." In *The European Conference on Computer Vision (ECCV)*, pp. 469–485, 2018.

[GMT22] Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. "Improving alignment of dialogue agents via targeted human judgements." *arXiv preprint arXiv:2209.14375*, 2022.

[GMZ23] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. "PAL: Program-aided language models." In *International Conference on Machine Learning (ICML)*, pp. 10764–10799. PMLR, 2023.

[Goo23] Google. "Bard.", 2023.

[GPS23] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. "Reinforced self-training (rest) for language modeling." *arXiv preprint arXiv:2308.08998*, 2023.

[GSG23] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. "ToRA: A tool-integrated reasoning agent for mathematical problem solving." *arXiv preprint arXiv:2309.17452*, 2023.

[GY18] Wenbin Gan and Xinguo Yu. "Automatic understanding and formalization of natural language geometry problems using syntax-semantics models." *International Journal of Innovative Computing, Information and Control*, pp. 83–98, 2018.

[GYZ19] Wenbin Gan, Xinguo Yu, Ting Zhang, and Mingshu Wang. "Automatically proving plane geometry theorems stated by text and diagram." *International Journal of Pattern Recognition and Artificial Intelligence*, **33**(07):1940003, 2019.

[HAD18] 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)*, pp. 53–69, 2018.

[HAP22] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents." In *International Conference on Machine Learning (ICML)*, pp. 9118–9147. PMLR, 2022.

[HAR17] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. "Learning to reason: End-to-end module networks for visual question answering." In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pp. 804–813, 2017.

[HBK21] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. "Measuring mathematical problem solving with the MATH dataset." In *The Conference on Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2021.

[HGJ19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. "Parameter-efficient transfer learning for NLP." In *International Conference on Machine Learning (ICML)*, pp. 2790–2799. PMLR, 2019.

[HLC21] Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. "Learning by fixing: Solving math word problems with weak supervision." In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, pp. 4959–4967, 2021.

[HLP19] Mark Hopkins, Ronan Le Bras, Cristian Petrescu-Prahova, Gabriel Stanovsky, Hannaneh Hajishirzi, and Rik Koncel-Kedziorski. "Semeval-2019 task 10: Math question answering." In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 893–899, 2019.

[HM19] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6700–6709, 2019.

[HQC24] Kaixuan Huang, Yuanhao Qu, Henry Cousins, William A Johnson, Di Yin, Mihir Shah, Denny Zhou, Russ Altman, Mengdi Wang, and Le Cong. "Crispr-GPT: An LLM agent for automated design of gene-editing experiments." *arXiv preprint arXiv:2404.18021*, 2024.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." *Neural computation*, **9**(8):1735–1780, 1997.

[HSL16] Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. "How well do computers solve math word problems? large-scale dataset construction and evaluation." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 887–896, 2016.

[HSL17] Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. "Learning fine-grained expressions to solve math word problems." In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pp. 805–814, 2017.

[HXX23] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. "Inner monologue: Embodied reasoning through planning with language models." In *Conference on Robot Learning*, pp. 1769–1782. PMLR, 2023.

[HZ05] Feng Han and Song-Chun Zhu. "Bottom-up/top-down image parsing by attribute graph grammar." In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pp. 1778–1785. IEEE, 2005.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[IDS23] Shima Imani, Liang Du, and Harsh Shrivastava. "MathPrompter: Mathematical reasoning using large language models." In *ICLR 2023 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models*, 2023.

[IYC17] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. “Search-based neural structured learning for sequential question answering.” In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1821–1831, 2017.

[Jai20] JaidedAI. “EasyOCR: Ready-to-use OCR.”, 2020.

[JC20] Harsh Jhamtani and Peter Clark. “Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering.” In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 137–150, 2020.

[JHM17] 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 (CVPR)*, pp. 2901–2910, 2017.

[JHV17] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. “Inferring and executing programs for visual reasoning.” In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pp. 2989–2998, 2017.

[JMR20] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. “In defense of grid features for visual question answering.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10267–10276, 2020.

[JTH16] Sujay Kumar Jauhar, Peter Turney, and Eduard Hovy. “TabMCQ: A dataset of general knowledge tables and multiple-choice questions.” *arXiv preprint arXiv:1602.03960*, 2016.

[JWM18] Peter Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton Morrison. “WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference.” In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, 2018.

[JYS20] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. “TinyBERT: Distilling BERT for natural language understanding.” In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, 2020.

[Kah11] Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.

[Kar11] Sevilay Karamustafaoğlu. “Improving the science process skills ability of science student teachers using I diagrams.” *International Journal of Physics & Chemistry Education*, **3**(1):26–38, 2011.

[KB14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization.” *arXiv preprint arXiv:1412.6980*, 2014.

[KBT16] Kirthevasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David Carter. "Batch policy gradient methods for improving neural conversation models." In *International Conference on Learning Representations (ICLR)*, 2016.

[KBY20] Takuro Karamatsu, Gibran Benitez-Garcia, Keiji Yanai, and Seiichi Uchida. "Iconify: Converting photographs into icons." In *Proceedings of the 2020 Joint Workshop on Multimedia Artworks Analysis and Attractiveness Computing in Multimedia*, pp. 7–12, 2020.

[KCC08] Terry Koo, Xavier Carreras, and Michael Collins. "Simple semi-supervised dependency parsing." In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 595–603, 2008.

[KCG20] Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. "QASC: A dataset for question answering via sentence composition." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 8082–8090, 2020.

[KCK21] Yannis Katsis, Saneem Chemmengath, Vishwajeet Kumar, Samarth Bharadwaj, Mustafa Canim, Michael Glass, Alfio Gliozzo, Feifei Pan, Jaydeep Sen, Karthik Sankaranarayanan, et al. "AIT-QA: Question answering dataset over complex tables in the airline industry." *arXiv preprint arXiv:2106.12944*, 2021.

[KGR22] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. "Large language models are zero-shot reasoners." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 22199–22213, 2022.

[KJZ18] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. "Bilinear attention networks." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1571–1581, 2018.

[KKR21] Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. "Text modular networks: Learning to decompose tasks in the language of existing models." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 1264–1279, 2021.

[KMA17] Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. "FigureQA: An annotated figure dataset for visual reasoning." *arXiv preprint arXiv:1710.07300*, 2017.

[KMK20] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. "UnifiedQA: Crossing format boundaries with a single QA system." In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1896–1907, 2020.

[KPC18] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. "DVQA: Understanding data visualizations via question answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5648–5656, 2018.

[KSK16] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. "A diagram is worth a dozen images." In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[KSK21] Wonjae Kim, Bokyung Son, and Ildoo Kim. "ViLT: Vision-and-language transformer without convolution or region supervision." In *International Conference on Machine Learning (ICML)*, pp. 5583–5594, 2021.

[KSS17] Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. "Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4999–5007, 2017.

[KSW22] Mojtaba Komeili, Kurt Shuster, and Jason Weston. "Internet-augmented dialogue generation." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 8460–8478, 2022.

[KT19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, p. 2, 2019.

[KTF23] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. "Decomposed prompting: A modular approach for solving complex tasks." In *International Conference on Learning Representations (ICLR)*, 2023.

[KTK16] Jayant Krishnamurthy, Oyvind Tafjord, and Aniruddha Kembhavi. "Semantic parsing to probabilistic programs for situated question answering." In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 160–170, 2016.

[KZG17] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. "Visual Genome: Connecting language and vision using crowdsourced dense image annotations." *International Journal of Computer Vision (IJCV)*, pp. 32–73, 2017.

[LBM21] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity." *arXiv preprint arXiv:2104.08786*, 2021.

[LBM22] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 8086–8098, 2022.

[LBP19] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. "ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 13–23, 2019.

[LBX24] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. "MathVista: Evaluating mathematical reasoning of foundation models in visual contexts." In *International Conference on Learning Representations (ICLR)*, 2024.

[LCG22] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. "TAPEX: Table pre-training via learning a neural sql executor." In *International Conference on Learning Representations (ICLR)*, 2022.

[LDC22] Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. "Can language models learn from explanations in context?" In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 537–563, 2022.

[LFG24] Michael Y Li, Emily B Fox, and Noah D Goodman. "Automated statistical model discovery with language models." *arXiv preprint arXiv:2402.17879*, 2024.

[LGG17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[LGG19] Manuel Lagunas, Elena Garces, and Diego Gutierrez. "Learning icons appearance similarity." *Multimedia Tools and Applications*, **78**(8):10733–10751, 2019.

[LGJ21] Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. "Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning." In *The 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[LHP15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971*, 2015.

[Lin04] Chin-Yew Lin. "ROUGE: A package for automatic evaluation of summaries." In *Text Summarization Branches Out*, pp. 74–81, 2004.

[LIS17] Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. "Deep network guided proof search." *arXiv preprint arXiv:1701.06972*, 2017.

[LJZ18] Pan Lu, Lei Ji, Wei Zhang, Nan Duan, Ming Zhou, and Jianyong Wang. "R-VQA: Learning visual relation facts with semantic attention for visual question answering." In *The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1880–1889, 2018.

[LKB23] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. “Let’s verify step by step.” In *International Conference on Learning Representations (ICLR)*, 2023.

[LLC23] Yujie Lu, Pan Lu, Zhiyu Chen, Wanrong Zhu, Xin Eric Wang, and William Yang Wang. “Multimodal procedural planning via dual text-image prompting.” *arXiv preprint arXiv:2305.01795*, 2023.

[LLG20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.” In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 7871–7880, 2020.

[LLW23] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. “Visual instruction tuning.” In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

[LLZ18] Pan Lu, Hongsheng Li, Wei Zhang, Jianyong Wang, and Xiaogang Wang. “Co-attending free-form regions and detections with multi-modal multiplicative feature embedding for visual question answering.” In *The AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[LMX22] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. “Learn to explain: Multimodal reasoning via thought chains for science question answering.” In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[LMZ19] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. “Large-scale long-tailed recognition in an open world.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2537–2546, 2019.

[LPC23] 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.” In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[LQC21] Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. “IconQA: A new benchmark for abstract diagram understanding and visual language reasoning.” In *The The Conference on Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2021.

[LQC23] Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. “Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning.” In *International Conference on Learning Representations (ICLR)*, 2023.

[LQY23] Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. "A survey of deep learning for mathematical reasoning." In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 14605–14631, 2023.

[LSF10] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. "Object bank: A high-level image representation for scene classification & semantic feature sparsification." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1378–1386, 2010.

[LST24] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander Rush, Douwe Kiela, et al. "OBELICS: An open web-scale filtered dataset of interleaved image-text documents." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.

[LSZ22] Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. "What makes good in-context examples for GPT-3?" In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114, 2022.

[LWW24] Xiao Liu, Zirui Wu, Xueqing Wu, Pan Lu, Kai-Wei Chang, and Yansong Feng. "Are LLMs capable of data-based statistical and causal reasoning? Benchmarking advanced quantitative reasoning with data." *arXiv preprint arXiv:2402.17644*, 2024.

[LXB24] Bo Lin, Yingjing Xu, Xuanwen Bao, Zhou Zhao, Zuyong Zhang, Zhouyang Wang, Jie Zhang, Shuiguang Deng, and Jianwei Yin. "SkinGEN: An explainable dermatology diagnosis-to-generation framework with interactive vision-language models." *arXiv preprint arXiv:2404.14755*, 2024.

[LYY19] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. "VisualBERT: A simple and performant baseline for vision and language." *arXiv preprint arXiv:1908.03557*, 2019.

[LYY20] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. "What does bert with vision look at?" In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5265–5275, 2020.

[LZR24] Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. "MathGenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of LLMs." *arXiv preprint arXiv:2402.16352*, 2024.

[Mar08] Maria Martiniello. "Language and the performance of English-language learners in math word problems." *Harvard Educational Review*, **78**(2):333–368, 2008.

[MBM16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning." In *International Conference on Machine Learning (ICML)*, pp. 1928–1937. PMLR, 2016.

[MBT18] Spandan Madan, Zoya Bylinskii, Matthew Tancik, Adrià Recasens, Kimberli Zhong, Sami Alsheikh, Hanspeter Pfister, Aude Oliva, and Fredo Durand. "Synthetically trained icon proposals for parsing and summarizing infographics." *arXiv preprint arXiv:1807.10441*, 2018.

[MCK18] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. "Can a suit of armor conduct electricity? a new dataset for open book question answering." *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[MCL22] Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. "Open domain question answering with a unified knowledge interface." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1605–1620, 2022.

[MDT22] Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. "ChartQA: A benchmark for question answering about charts with visual and logical reasoning." In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279, 2022.

[MFL22] Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. "Lila: A unified benchmark for mathematical reasoning." In *The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[MII17] Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H Arai. "Semantic parsing of pre-university math problems." In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2131–2141, 2017.

[MKB21] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. "Cross-task generalization via natural language crowdsourcing instructions." In *The 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[MKB22] Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. "Reframing instructional prompts to GPTk's language." In *Findings of the Association for Computational Linguistics (ACL)*, pp. 589–612, 2022.

[MLS20] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. "A diverse corpus for evaluating and developing English math word problem solvers." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 975–984, 2020.

[MTG24] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. "Self-refine: Iterative refinement with self-feedback." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.

[MYZ22] Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. "SQA3D: Situated question answering in 3d scenes." In *International Conference on Learning Representations (ICLR)*, 2022.

[NAG22] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. "Show your work: Scratchpads for intermediate computation with language models." In *Deep Learning for Code Workshop*, 2022.

[NHB21] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. "WebGPT: Browser-assisted question-answering with human feedback." *arXiv preprint arXiv:2112.09332*, 2021.

[NHM22] Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Nick Schoelkopf, Riley Kong, Xiangru Tang, et al. "FeTaQA: Free-form table question answering." *Transactions of the Association for Computational Linguistics (TACL)*, **10**:35–49, 2022.

[NN17] Andi Saparuddin Nur and Evy Nurvitasari. "Geometry skill analysis in problem solving reviewed from the difference of cognitive style students junior high school." *Journal of Educational Science and Technology*, **3**(3):204–210, 2017.

[NRL20] Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. "Wt5?! training text-to-text models to explain their predictions." *arXiv preprint arXiv:2004.14546*, 2020.

[Ope22] OpenAI. "ChatGPT.", 2022.

[Ope23a] OpenAI. "GPT-4 technical report." *arXiv preprint arXiv:2303.08774*, 2023.

[Ope23b] OpenAI. "GPT-4V(ision) system card.", 2023.

[OWJ22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. "Training language models to follow instructions with human feedback." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 27730–27744, 2022.

[PBG21] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. "Are NLP models really able to solve simple math word problems?" In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 2080–2094, 2021.

[PCS20] Rachel Pollitt, Caroline Cohrssen, and Wee Tiong Seah. "Assessing spatial reasoning during play: Educator observations, assessment and curriculum planning." *Mathematics Education Research Journal*, **32**(2):331–363, 2020.

[PGH23] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. "Check your facts and try again: Improving large language models with external knowledge and automated feedback." *arXiv preprint arXiv:2302.12813*, 2023.

[PKC21] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. "True few-shot learning with language models." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[PL15] Panupong Pasupat and Percy Liang. "Compositional semantic parsing on semi-structured tables." In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJNLP)*, pp. 1470–1480, 2015.

[PLH23] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. "Instruction tuning with GPT-4." *arXiv preprint arXiv:2304.03277*, 2023.

[PLS23] Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. "ART: Automatic multi-step reasoning and tool-use for large language models." *arXiv preprint arXiv:2303.09014*, 2023.

[PMP22] Mihir Parmar, Swaroop Mishra, Mirali Purohit, Man Luo, Murad Mohammad, and Chitta Baral. "In-BoXBART: Get instructions into biomedical multi-task learning." In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 112–128, 2022.

[PRW02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: A method for automatic evaluation of machine translation." In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 311–318, 2002.

[PS06] Jan Peters and Stefan Schaal. "Policy gradient methods for robotics." In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2219–2225. IEEE, 2006.

[PSD18] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. "FiLM: Visual reasoning with a general conditioning layer." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 32, 2018.

[QLL20] Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. "Semantically-aligned universal tree-structured solver for math word problems." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3780–3789, 2020.

[QZL22] Liang Qiu, Yizhou Zhao, Jinchao Li, Pan Lu, Baolin Peng, Jianfeng Gao, and Song-Chun Zhu. "ValueNet: A new dataset for human value driven dialogue system." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 11183–11191, 2022.

[RBN23] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. "Mathematical discoveries from program search with large language models." *Nature*, pp. 1–3, 2023.

[RG19] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence embeddings using siamese bert-networks." 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)*, pp. 3982–3992, 2019.

[RHG15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 91–99, 2015.

[Ric23] Toran Bruce Richards. "Auto-GPT: An experimental open-source attempt to make GPT-4 fully autonomous." `https://github.com/Significant-Gravitas/Auto-GPT`, 2023.

[RR18] Subhro Roy and Dan Roth. "Mapping to declarative knowledge for word problem solving." *Transactions of the Association for Computational Linguistics (TACL)*, **6**:159–172, 2018.

[RSR20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer." *Journal of Machine Learning Research (JMLR)*, **21**:1–67, 2020.

[RWC20] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners." *OpenAI Blog*, 2020.

[SB98] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.

[SCA23] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, et al. "Beyond human data: Scaling self-training for problem-solving with language models." *arXiv preprint arXiv:2312.06585*, 2023.

[SDD23] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. "Toolformer: Language models can teach themselves to use tools." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

[SDH20] Mrinmaya Sachan, Avinava Dubey, Eduard H Hovy, Tom M Mitchell, Dan Roth, and Eric P Xing. "Discourse in multimedia: A case study in extracting geometry knowledge from textbooks." *Computational Linguistics*, **45**(4):627–665, 2020.

[SDM18] Mrinmaya Sachan, Kumar Avinava Dubey, Tom M Mitchell, Dan Roth, and Eric P Xing. "Learning pipelines with limited data and domain knowledge: A study in parsing physics problems." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 140–151, 2018.

[SDX17] Mrinmaya Sachan, Kumar Dubey, and Eric Xing. "From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems." In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pp. 773–784, 2017.

[SFA22] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. "BLOOM: A 176b-parameter open-access multilingual language model." *arXiv preprint arXiv:2211.05100*, 2022.

[SHF14] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. "Diagram understanding in geometry questions." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[SHF15] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. "Solving geometry problems: Combining text and diagram interpretation." In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1466–1476, 2015.

[SHZ24] Liangtai Sun, Yang Han, Zihan Zhao, Da Ma, Zhennan Shen, Baocai Chen, Lu Chen, and Kai Yu. "SciEval: A multi-level large language model evaluation benchmark for scientific research." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 19053–19061, 2024.

[SI89] Deborah Stipek and Douglas Mac Iver. "Developmental change in children's assessment of intellectual competence." *Child development*, pp. 521–538, 1989.

[SLH14] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. "Deterministic policy gradient algorithms." In *International Conference on Machine Learning (ICML)*, pp. 387–395. PMLR, 2014.

[SLY17] Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. "A corpus of natural language for visual reasoning." In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 217–223, 2017.

[SMV23] Dídac Surís, Sachit Menon, and Carl Vondrick. "ViperGPT: Visual inference via python execution for reasoning." In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11888–11898, 2023.

[SN12] Mike Schuster and Kaisuke Nakajima. "Japanese and korean voice search." In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152. IEEE, 2012.

[SNS19] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. "Towards VQA models that can read." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8317–8326, 2019.

[SS01] Linda G Shapiro and George C Stockman. *Computer Vision*. Prentice Hall, 2001.

[SST23] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. "HuggingGPT: Solving ai tasks with ChatGPT and its friends in Hugging Face." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, 2014.

[SWD17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347*, 2017.

[SX17] Mrinmaya Sachan and Eric Xing. "Learning to solve geometry problems from natural language demonstrations in textbooks." In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, pp. 251–261, 2017.

[SYB20] Shailaja Keyur Sampat, Yezhou Yang, and Chitta Baral. "Visuo-linguistic question answering (VLQA) challenge." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings (EMNLP)*, pp. 4606–4616, 2020.

[TAB23] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. "Gemini: A family of highly capable multimodal models." *arXiv preprint arXiv:2312.11805*, 2023.

[TCL19] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Well-read students learn better: On the importance of pre-training compact models." *arXiv preprint arXiv:1908.08962*, 2019.

[Tea24] Gemini Team. "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.", 2024.

[TKC22] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. "Galactica: A large language model for science." *arXiv preprint arXiv:2211.09085*, 2022.

[TLI23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "LLaMA: Open and efficient foundation language models." *arXiv preprint arXiv:2302.13971*, 2023.

[TML14] Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song-Chun Zhu. "Joint video and text parsing for understanding events and answering queries." *IEEE MultiMedia*, **21**(2):42–70, 2014.

[TMS23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. "LLaMA 2: Open foundation and fine-tuned chat models." *arXiv preprint arXiv:2307.09288*, 2023.

[TWL24] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. "Solving olympiad geometry without human demonstrations." *Nature*, **625**(7995):476–482, 2024.

[TYC20] Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. "MultiModalQA: Complex question answering over text, tables and images." In *International Conference on Learning Representations (ICLR)*, 2020.

[UC17] Shyam Upadhyay and Ming-Wei Chang. "Annotating derivations: A new evaluation strategy and dataset for algebra word problems." In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (ACL)*, pp. 494–504, 2017.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5998–6008, 2017.

[WBZ21] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. "Finetuned language models are zero-shot learners." *The International Conference on Learning Representations (ICLR)*, 2021.

[Wen86] Wu Wen-Tsun. "Basic principles of mechanical theorem proving in elementary geometries." *Journal of Automated Reasoning*, **2**(3):221–252, 1986.

[Wil92] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning*, **8**(3):229–256, 1992.

[WIL23] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. "BloombergGPT: A large language model for finance." *arXiv preprint arXiv:2303.17564*, 2023.

[WLJ22] Xingyao Wang, Sha Li, and Heng Ji. "Code4Struct: Code generation for few-shot structured prediction from natural language." *arXiv preprint arXiv:2210.12810*, 2022.

[WLS17] Yan Wang, Xiaojiang Liu, and Shuming Shi. "Deep neural solver for math word problems." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 845–854, 2017.

[WLS20] Xinyu Wang, Yuliang Liu, Chunhua Shen, Chun Chet Ng, Canjie Luo, Lianwen Jin, Chee Seng Chan, Anton van den Hengel, and Liangwei Wang. "On the general value of evidence, and bilingual scene-text visual question answering." In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[Won21] Ildoo Kim Wonjae Kim, Bokyung Son. "An image is worth 16x16 words: Transformers for image recognition at scale." In *International Conference on Learning Representations (ICLR)*, 2021. under review.

[WRZ23] Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. "MathCoder: Seamless Code Integration in LLMs for Enhanced Mathematical Reasoning." In *International Conference on Learning Representations (ICLR)*, 2023.

[WTB22] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. "Emergent abilities of large language models." *Transactions on Machine Learning Research*, 2022.

[WWC18] Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. "Translating a math word problem to an expression tree." In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[WWS22a] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. "Self-consistency improves chain of thought reasoning in language models." In *International Conference on Learning Representations (ICLR)*, 2022.

[WWS22b] 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." In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24824–24837, 2022.

[WWZ15] Shuo Wang, Yizhou Wang, and Song-Chun Zhu. "Learning hierarchical space tiling for scene modeling, parsing and attribute tagging." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(12):2478–2491, 2015.

[WYQ23] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. "Visual ChatGPT: Talking, drawing and editing with visual foundation models." *arXiv preprint arXiv:2303.04671*, 2023.

[WZZ19] Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. "Template-based math word problem solvers with recursive neural networks." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pp. 7144–7151, 2019.

[YD19] Kaiyu Yang and Jia Deng. "Learning to prove theorems via interacting with proof assistants." In *International Conference on Machine Learning (ICML)*, pp. 6984–6994. PMLR, 2019.

[YIW23] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. “Generate rather than retrieve: Large language models are strong context generators.” In *International Conference on Learning Representations (ICLR)*, 2023.

[YLW23a] Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. “FinGPT: Open-source financial large language models.” *arXiv preprint arXiv:2306.06031*, 2023.

[YLW23b] 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.

[YWG18] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. “Neural-symbolic VQA: Disentangling reasoning from vision and language understanding.” In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[YWG19] Xinguo Yu, Mingshu Wang, Wenbin Gan, Bin He, and Nan Ye. “A framework for solving explicit arithmetic word problems and proving plane geometry theorems.” *International Journal of Pattern Recognition and Artificial Intelligence*, **33**(07):1940005, 2019.

[YXX23] Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. “mPLUG-Owl: Modularization empowers large language models with multimodality.” *arXiv preprint arXiv:2304.14178*, 2023.

[YYC19] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. “Deep modular co-attention networks for visual question answering.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6281–6290, 2019.

[YZY18] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. “Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task.” In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3911–3921, 2018.

[ZCS23] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. “MiniGPT-4: Enhancing vision-language understanding with advanced large language models.” In *International Conference on Learning Representations (ICLR)*, 2023.

[ZGB16] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. “Visual7W: Grounded question answering in images.” In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[ZGS16] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. “Yin and Yang: Balancing and answering binary visual questions.” In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[ZHZ24] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. “LLaMA-Adapter: Efficient fine-tuning of language models with zero-init attention.” In *International Conference on Learning Representations (ICLR)*, 2024.

[ZLH21] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. “TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance.” In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-JCNLP)*, pp. 3277–3287, 2021.

[ZLL22] Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. “MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data.” In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 6588–6600, 2022.

[ZM06] Song-Chun Zhu and David Mumford. “A stochastic grammar of images.” *Foundations and Trends in Computer Graphics and Vision*, **2**(4):259–362, 2006.

[ZWF21] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. “Calibrate before use: Improving few-shot performance of language models.” In *International Conference on Machine Learning (ICML)*, pp. 12697–12706. PMLR, 2021.

[ZWL23] Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. “Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification.” In *International Conference on Learning Representations (ICLR)*, 2023.

[ZWZ15] Jun Zhu, Tianfu Wu, Song-Chun Zhu, Xiaokang Yang, and Wenjun Zhang. “A reconfigurable tangram model for scene representation and categorization.” *IEEE Transactions on Image Processing*, **25**(1):150–166, 2015.

[ZXS17] Victor Zhong, Caiming Xiong, and Richard Socher. “Seq2SQL: Generating structured queries from natural language using reinforcement learning.” *arXiv preprint arXiv:1709.00103*, 2017.

[ZZG23] 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.

[ZZL23] 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.