Geometric Question Answering Towards Multimodal Numerical Reasoning

Jiaqi Chen[1*], **Jianheng Tang**[2*], **Jinghui Qin**[1], **Xiaodan Liang**[2†], **Lingbo Liu**[1,3], **Eric P. Xing**[4], **Liang Lin**[1,3]
[1]Sun Yat-sen University, [2]Shenzhen Campus of Sun Yat-sen University, [3]Dark Matter AI Inc.,
[4]Mohamed bin Zayed University of Artificial Intelligence

Abstract

Automatic math problem solving has recently attracted increasing attention as a long-standing AI benchmark. In this paper, we focus on solving geometric problems, which requires a comprehensive understanding of textual descriptions, visual diagrams, and theorem knowledge. However, the existing methods were highly dependent on handcraft rules and were merely evaluated on small-scale datasets. Therefore, we propose a **Geo**metric **Q**uestion **A**nswering dataset **GeoQA**, containing 5,010 geometric problems with corresponding annotated programs, which illustrate the solving process of the given problems. Compared with another publicly available dataset GeoS, GeoQA is 25 times larger, in which the program annotations can provide a practical testbed for future research on explicit and explainable numerical reasoning. Moreover, we introduce a Neural Geometric Solver (NGS) to address geometric problems by comprehensively parsing multimodal information and generating interpretable programs. We further add multiple self-supervised auxiliary tasks on NGS to enhance cross-modal semantic representation. Extensive experiments on GeoQA validate the effectiveness of our proposed NGS and auxiliary tasks. However, the results are still significantly lower than human performance, which leaves large room for future research.

1 Introduction

In recent years, developing machine learning systems to solve math word problems (MWPs) automatically has attracted increasing attention due to its high academic value and the great application potential in smart education [1, 2]. Most of the existing methods focus on solving arithmetic and algebraic problems [3, 4, 5, 6, 7, 8] , while solving geometric problems has been rarely investigated [9, 10, 11]. As a classic math problem, geometry dominates a large portion of secondary education. Due to its challenges and data characteristics, geometry problem can also serve as a multimodal numerical reasoning benchmark requiring joint reasoning over diagram and text.

A typical geometric problem is shown in Figure 1. Compared with math word problems, geometric questions have posed the following new challenges. **First**, the additional problem diagrams provide essential information absent from the problem text, such as the relative location of lines and points; thus, the solver should have the capability to parse the diagram. **Second**, to solve a geometry problem, we need to understand and align the semantics of text and diagram simultaneously. However, the problem text often includes some ambiguous references and implicit relations to diagram elements, which increases the difficulty of joint reasoning over text and diagram. **Third**, many geometric problems require extra theorem knowledge in the problem solving process. For example, in Figure 1, the Pythagorean Theorem is used to calculate the length of line AE.

[*] Equal contribution. † Corresponding author. Benchmark and code: https://github.com/chen-judge/GeoQA

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Workshop on Math AI for Education (MATHAI4ED).

Though some previous methods [9, 10, 11, 12, 13] attempt to resolve the mentioned issues, the performance of their geometric problem solving systems is far away from satisfactory. They highly depended on limited handcraft rules and were only validated on small-scale datasets, making it hard to generalize to more complex and real-world cases. Besides, it is difficult for human to understand and examine the reliability of the sophisticated solving process.

Therefore, we propose a large-scale real-world geometric question answering dataset called GeoQA, which contains 5,010 multiple-choice geometric problems collected from real math exams in Chinese middle school. Inspired by [14], we additionally introduce a new domain-specific language to model precise operation programs corresponding to the geometry problem. These executable programs represent the numerical reasoning steps of geometry problems. Compared with the existing dataset GeoS and GeoS++ [10, 11], our GeoQA is larger, more diverse, provides additional program annotation, thus serves as a promising benchmark to improve both generalization and interpretability of the multimodal numerical reasoning approaches.

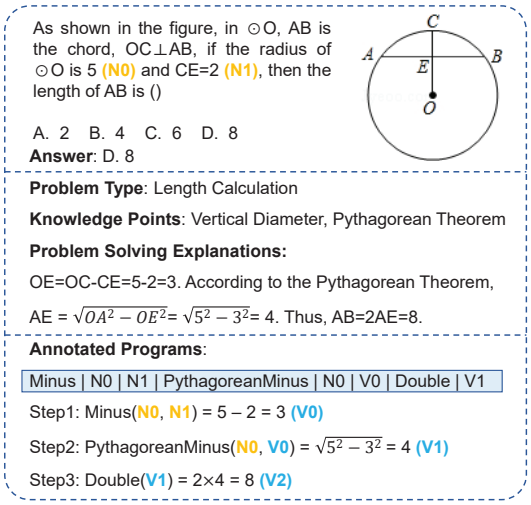

Figure 1: Illustration of a typical geometry problem with the annotated programs in our GeoQA dataset.

Moreover, we propose the first deep learning-based approach for geometry problem solving, named as Neural Geometric Solver (NGS). It applies a co-attention mechanism to fuse the representation of text and diagram, and predicts the explainable programs based on the cross-modal representation. These sequential programs can be executed to obtain a final answer. We further design three highly-relevant pretext tasks to enhance text-diagram semantic representation. Extensive experiments are conducted on GeoQA benchmark, and the quantitative comparisons show the superiority of the proposed NGS and auxiliary tasks over other multimodal baselines.

2 GeoQA Dataset

We collect a new dataset GeoQA, containing 5,010 diverse real-world geometric problems in Chinese middle school exams. Each problem is additionally annotated by specific programs that describe the problem solving process. See Appendix A for more details of the GeoQA.

2.1 Data Description

There are three problem types in our GeoQA, i.e., angle calculation, length calculation, and others which contain various types of problems such as area calculation. We adopt the corpus diversity metric proposed by [15] to evaluate the diversity of GeoQA. The result is 0.47, which is relatively high compared with other math problem datasets, indicating that our dataset is diverse. The source data already contains manually tagged knowledge points in each problem, we design rule-based regular expressions to normalize the original knowledge points to 50 categories. We split our GeoQA into three subsets – train set, valid set, and test set, in a ratio of 7.0: 1.5: 1.5. The data statistics of our GeoQA are shown in Table 1.

	Total	Train	Val	Test
Number	5010	3509	746	755
Angle	2745	1939	388	418
Length	1873	1303	287	283
Other	392	267	71	54
#Avg DS	108×140	108×140	107×141	107×140
#Avg QL	52.5	52.4	52.4	57.7
#Avg KP	2.10	2.10	2.07	2.14
#Avg ET	1.11	1.13	1.08	1.09
#Avg OP	1.98	1.99	1.92	1.98
#Avg PL	5.35	5.39	5.17	5.36

Table 1: Statistics of our GeoQA dataset with three types of problems. DS, QL, KP, and ET represent diagram size, question length, knowledge points, and element types, respectively. OP and PL represent operation number and program length.

2.2 Program Representation

To make better use of neural networks in the geometric problems solving process, inspired by [14], we introduce a new domain-specific language to model the geometric problem solving process based on the GeoQA dataset. This program language can be directly executed to calculate the numerical answer based on the predefined operations and their arguments.

The program types contain operations OP, constants $Const$, problem variables N, and process variables V. As shown in Table 2, operations are divided into multiple categories, including Basic, Arithmetic, Trigonometric, Theorem, and Formula operations. Each operator involves $n(= 1, 2, 3)$ elements selected from constants and variables. Constants are predefined numbers that are frequently used in geometric problems, such as π and the degree of a Right Angle (90). The problem variables refer to all the number that appears in the current problem, and process variables are obtained during the execution process.

Types	Programs
Basic	Equal, Double, Half
Arithmetic	Add, Minus, Multiply, Divide
Trigonometric	Sin, Cos, Tan, Arc-Sin, Arc-Cos
Theorem & Formula	Pythagorean Add/Minus, Proportion, Circle Area, Circle Perimeter, Cone Area
Constant	30, 60, 90, 180, 360, π, 0.618

Table 2: An overview of 18 operations of four different types and 7 constants in the defined program set.

In addition to the common math operations, our programs also contain some operations representing the knowledge of theorems and formulas that is helpful to address geometric problems, such as the Pythagorean theorem and the area calculation formula of a circle. It is worth noting that many simple geometric formulas do not require additional definitions. For example, given a square with side length a, its area can be directly computed by Multiply(a, a).

The interpretability of our program is reflected on the sequential process of the operations, the selected constants and variables, and the application of theorems and formulas. As shown in Figure 1, we can have a general understanding of the entire problem solving process after reading the program.

3 Neural Geometric Solver

We propose Neural Geometric Solver (NGS) to address geometric problems by jointly understanding text, diagram, and then generating explainable programs. Moreover, we utilize some novel auxiliary tasks to enhance the understanding ability of our NGS. The overall architecture of our NGS is shown in Fig. 2. The problem text and diagram are encoded separately, then fed into a joint reasoning module together to obtain cross-modal fusion of text and diagram features. A decoder utilizes fused multimodal features to generate interpretable programs. In addition, the proposed three auxiliary tasks contain jigsaw location prediction, geometric elements prediction, and knowledge points prediction, all of which enhance feature representation and facilitate multimodal reasoning. View Appendix B for more details about the components of NGS network.

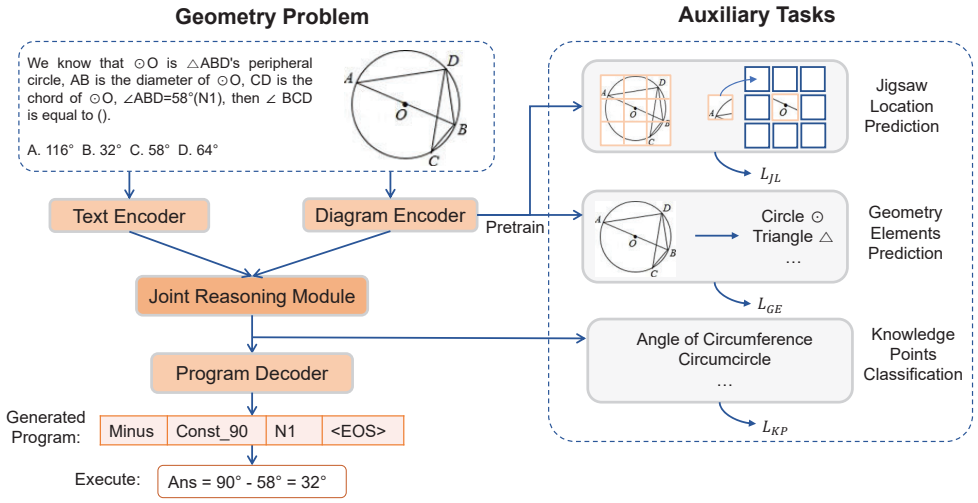

Figure 2: The overall architecture of our Neural Geometric Solver in conjunction with auxiliary tasks.

Method		Total (%)	Angle (%)	Length (%)	Other (%)
Human	Text-Only	63.0	58.1	71.7	55.6
	Text-Diagram	92.3	94.3	90.5	87.0
W/O Program	FiLM [16]	31.7	34.0	29.7	24.1
	RN [17]	38.0	42.8	32.5	29.6
	MCAN [18]	39.7	45.0	34.6	25.9
Text-Only	Seq2Prog [14]	52.3	62.4	42.1	27.8
	BERT2Prog [19]	54.7	65.8	42.1	35.2
Text-Diagram	BERT2Prog + Diagram	50.3	63.4	33.2	38.9
	Seq2Prog + Diagram	52.6	63.6	39.2	37.0
	NGS (Ours)	56.7	67.5	44.5	37.0
	NGS-Auxiliary (Ours)	**60.7**	**72.0**	**47.0**	**44.4**

Table 3: The answer accuracy comparison on different test subsets of GeoQA dataset.

4 Experiments

We introduce three types of models here and test them on our GeoQA. The performance comparison with various methods on the different subset types of GeoQA is reported in Table 3. We provide more experimental details and analysis in Appendix C.

The effectiveness of program. "W/O Program" refers to not using programs and regarding GeoQA as a classification problem similar to VQA. Therefore, we conduct experiments on three models with multimodal reasoning capabilities, including FiLM, RN, and MCAN. However, their performance results show that these methods fail to reason about such complex geometric problems, achieving poor performance on GeoQA. These results prove the effectiveness and importance of our designed interpretable programs.

The necessity of multi-modality. "Text-Only" means that models only use text to generate program sequences since humans can also understand the intent of the question, draw diagram based on the text, and solve the problem. Motivated by [14], we design a Sequence-to-Program (Seq2Prog) model using GRU encoder with an attention mechanism. Moreover, we replace the encoder with BERT and get a stronger BERT2Prog. By predicting our tailor-designed programs, these two methods are effective, while the performance is not satisfactory enough. These results show that multimodal reasoning ability is indispensable when solving geometric problems.

Multimodal numerical reasoning baselines. "Text-Diagram" refers to using text and diagram simultaneously. We concatenate the text embedding with the diagram feature extracted by ResNet [20] as the baseline methods, such as Seq2Prog + Diagram and BERT2Prog + Diagram. These feature fusion methods do not have the strong reasoning ability and fail to improve the program decoding. For instance, by adding diagram, the performance of BERT2Prog + Diagram declines from 54.7% to 50.3%, which may result from the extra diagram that disturbs the text pretraining model.

The effectiveness of our methods. Our proposed NGS shows a relatively-good performance compared to the various models mentioned above. When adding all three auxiliary tasks together to enhance NGS solver, our NGS-Auxiliary with multimodal reasoning ability becomes the existing best-performing method (60.7%) on GeoQA dataset. It also achieves the highest accuracy on all types of problems. For example, compared with Seq2Prog+Diagram, NGS-Auxiliary obtains an 8.4% performance improvement on the angle type problems. Compared with other "Text-Diagram" baselines, our model is effective when reasoning on multimodal information.

5 Conclusion

In this work, we focus on the geometric problem and propose the first large-scale geometric question answering dataset "GeoQA", containing 5,010 problems with program annotation. Besides, we propose a deep neural baseline, named as Neural Geometric Solver (NGS), to solve a geometric problem by jointly reasoning over multimodal data and generating interpretable programs. We further propose multiple novel auxiliary tasks to enhance the semantic representation of text and diagram. Extensive experimental results and analyses show that our GeoQA is challenging, and our NGS-Auxiliary outperforms other methods on GeoQA.

References

[1] Richa Bajaj and Vidushi Sharma. Smart education with artificial intelligence based determination of learning styles. *Procedia computer science*, 132:834–842, 2018.

[2] Jinjiao Lin, Haitao Pu, Yibin Li, and Jian Lian. Intelligent recommendation system for course selection in smart education. *Procedia Computer Science*, 129:449–453, 2018.

[3] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 271–281, 2014.

[4] Lipu Zhou, Shuaixiang Dai, and Liwei Chen. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822. Association for Computational Linguistics, 2015.

[5] 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 (Volume 1: Long Papers)*, pages 887–896. Association for Computational Linguistics, 2016.

[6] Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854. Association for Computational Linguistics, 2017.

[7] Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069. Association for Computational Linguistics, 2018.

[8] Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization, 2019.

[9] Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. Diagram understanding in geometry questions. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[10] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476, 2015.

[11] 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 the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 773–784, 2017.

[12] 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.

[13] Mrinmaya Sachan. Knowledge graph embedding compression. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2681–2691, 2020.

[14] 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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2357–2367, 2019.

[15] 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*, pages 975–984, 2020.

[16] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017.

[17] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[18] 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*, pages 6281–6290, 2019.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[23] Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. An introduction to java geometry expert. In *International Workshop on Automated Deduction in Geometry*, pages 189–195. Springer, 2008.

[24] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476. Association for Computational Linguistics, 2015.

[25] 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 (* SEM 2017)*, pages 251–261, 2017.

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Herbert Gelernter, James R Hansen, and Donald W Loveland. Empirical explorations of the geometry theorem machine. In *Papers presented at the May 3-5, 1960, western joint IRE-AIEE-ACM computer conference*, pages 143–149, 1960.

[29] Wu Wen-Tsun. Basic principles of mechanical theorem proving in elementary geometries. *Journal of automated Reasoning*, 2(3):221–252, 1986.

[30] 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.

[31] 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. *arXiv preprint arXiv:2105.04165*, 2021.

[32] Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric P Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. *arXiv preprint arXiv:2105.14517*, 2021.

[33] 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*, pages 6904–6913, 2017.

[34] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[35] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *arXiv preprint arXiv:1810.02338*, 2018.

[36] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *International Conference on Learning Representations*, 2019.

[37] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

[38] Alejandro Newell and Jia Deng. How useful is self-supervised pretraining for visual tasks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7354, 2020.

[39] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[40] Unaiza Ahsan, Rishi Madhok, and Irfan Essa. Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 179–189. IEEE, 2019.

[41] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[42] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

A GeoQA

A.1 Dataset Collection and Annotation

We collect GeoQA from two online education websites[2]. These problems are oriented grades 6-12, containing various types of problems with corresponding knowledge points and solving explanations. we organize more than ten well-trained college students with a relevant major to specifically annotate our programs by referring to the solving explanations. To ensure label quality and consistency, they are required to read the guideline of annotation standards and examples in advance. Each annotated program is double-checked by one of the authors, and the annotator with low accuracy would be disqualified. The annotated operations required to solve the problem are limited to a maximum of 4 steps, thus a small number of complex and hard problems are filtered.

A.2 Human Performance

We invite 10 students with a high score (top 1%) in the national university entry exam to answer these geometric problems. The results are shown at the top of Table 3. For each question, they first try to solve the problem with the text only and draw a diagram by themselves. Then, the actual diagram is given to answer the complete question. The total time for each question is limited to two minutes. When using text and diagram simultaneously, the human performance is improved from 63.2% to 92.3%, which indicates that humans are not good at solving problems using only text, but handling multimodal information successfully. The result shows that there is still a huge gap between the existing models and human experts, leaving large room for future research.

B Details of NGS

B.1 The Architecture of NGS

B.1.1 Problem Encoder

Text Encoder Given a problem text $P = \{x_i\}_{i=1}^{n}$, each token x_i is first embedded into a word vector $\mathbf{x}_i$. A single-layer unidirectional LSTM [21] is then applied to encode each word embedding $\mathbf{x}_i$ into a hidden state $\mathbf{h}_i$. The sequence of the hidden state in LSTM are used to represent problem text P as $H_P = [\mathbf{h}_0; ...; \mathbf{h}_n]$.

Diagram Encoder For representing a problem diagram, we apply the first three stages of a ResNet-101 [20] to extract it as the diagram feature, which can be formalized as a feature matrix $H_D \in \mathbb{R}^{m \times d}$. Moreover, we also apply multiple auxiliary tasks for pretraining the diagram encoder. Note that the parameters of the diagram encoder are fixed when training the overall NGS.

B.1.2 Joint Reasoning Module

Given the text feature H_P and the diagram feature H_D, it is crucial for solving geometric problems to jointly fuse and align the cross-modal information. To this end, inspired by [18], we adopt a co-attention module to conduct cross-modal joint reasoning with attention mechanism. The co-attention module consists of 12 self-attention (SA) units and 6 guided-attention (GA) units, which fully fuse and align the text representation H_P and the diagram representation H_D. H_P is first encoded by 6 self-attention units (i.e., original Transformer), and the final hidden state processed by the 6-th self-attention unit is used as guiding information. Then, the guiding information is fed into another stacked 6 self-attention units and 6 guided-attention units to achieve cross-modal semantic fusion and alignment. Finally, the co-attention module outputs a cross-modal representation $F_D = [f_1^D; ...; f_n^D]$, which contains rich information over the problem text and diagram.

In this work, we find that text information is more fundamental than diagram information. Therefore, we further enhance the cross-modal representation with the help of textual information. Specifically, we concatenate H_P and F_D to acquire an enhanced reasoning module output F_R for decoding programs.

[2]http://www.zxxk.com/ and http://www.jyeoo.com/

Besides, an attentional reduction network with a two-layer MLP is applied to aggregate feature F_D into $\tilde{F}_D$. Similarly, we concatenate $\tilde{F}_D$ and the last encoder state of the text encoder $\boldsymbol{h}_n$, obtaining $\tilde{F}_R$ as the final gathered multimodal feature vector.

B.1.3 Program Decoder

The program decoder module generates the programs sequentially under the guidance of multimodal information. Concretely, we use a LSTM decoder [21] with attention [22] over the reasoning module output F_R . Let $\{y_t\}(1 \leq t \leq T)$ be the target program to be generated and $\boldsymbol{s}_t$ be the hidden state of LSTM at time step t. $\tilde{F}_R$ is fed into a linear layer to obtain the initial state $\boldsymbol{s}_0$. $\boldsymbol{s}_t$ is concatenated with the attention result and fed to a linear layer with the softmax function to predict the distribution of the next program token P_t.

During training, the generation loss L_g is the negative log-likelihood (NLL) of the target program:

$$\mathcal{L}_g(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} \log P_t(y_t|\boldsymbol{x}, y_1, ..., y_{t-1}; \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ are the parameters of the entire NGS architecture except for the diagram encoder, $\boldsymbol{x}$ is the input of both problem text and the extracted diagram feature. When testing, the decoder only observes the input text and diagram feature along with the program parts that have been generated.

B.1.4 Program Executor

After a beam of top N program sequences $\{g_1, ..., g_n\}$ are generated from the program decoder, the executor computes them step by step. When executing the program, the token sequence is first divided into several parts based on the position of operators in the program. Once a complete operation program has been decoded, each operator in the program is executed sequentially to obtain a numerical result. The execution process fails if g_i has a grammar error (e.g., the number of augments does not match the current operator), or the executed value does not match any options in the current problem. NGS adopts the first successfully executed program as the predicted solution and chooses the corresponding operation. If all N program sequences fail, the executor will report "no result" directly instead of guessing an option. Fig. 1 shows the detailed step-by-step execution of a final predicted program sequence.

B.2 Auxiliary Tasks

B.2.1 Self-supervised Diagram Auxiliary Task

Although our NGS can jointly fuse text feature and diagram feature with a co-attention mechanism, a powerful diagram encoder is needed to improve the problem understanding and answer accuracy. To obtain a high-quality diagram feature, we investigate two self-supervised auxiliary tasks, named as **Jigsaw Location Prediction** and **Geometry Elements Prediction**, to pretrain diagram encoder.

Jigsaw Location Prediction In the Jigsaw location prediction task that enforces pixel-level perception, we first split a diagram as $m \times m$ blocks and select the center block as the target. Then, we shuffle other blocks randomly and train the diagram encoder to predict the correct relative location between these shuffled blocks and the target using a cross-entropy loss.

Geometry Elements Prediction For object-level understanding, we design a geometry elements prediction task that aims at training the diagram encoder to predict the geometry elements appearing in the diagram. A diagram usually contains multiple geometry elements which are also mentioned in the problem text and the solving explanation. We extract these geometry elements from text as the label and deploy an N-way classifier with binary cross-entropy (BCE) as the loss function to train the diagram encoder, where N is the number of the possible geometry elements on the training set.

B.2.2 Knowledge Points Prediction

In addition to self-supervised diagram training, we also propose another auxiliary learning task called knowledge points prediction to enhance a problem's overall representation by providing an extra training signal. We summarize about 50 knowledge points for our GeoQA, and label each problem

with one or more knowledge points. We predict the knowledge points for each problem based on the gathered feature vector $\tilde{F}_R$ outputted from the joint reasoning module. Different from the diagram training, the knowledge points prediction task is trained with NGS simultaneously. We also deploy a K-way classifier with binary cross-entropy (BCE) as the loss function to train the knowledge points prediction multi-label task, where K is the total number of the possible knowledge points on the training set.

C Experimental Details and Analysis

C.1 Setup and Training Details

We conduct experiments on GeoQA dataset, and adopt answer accuracy as the evaluation metric. Although there is another available geometric problem dataset [10], the limited data scale (with only 67 training samples) makes it impossible to support neural network training. On the other hand, previous geometry problem solving systems require additional inputs (e.g., OCR and dependency parsing results) of the problem [23, 24] or not release their codes [25, 13]. Therefore, they are not comparable on our GeoQA dataset.

In this work, we implement the proposed method with Pytorch [26]. The learning rate is $1e^{-3}$ and the batch size is set to 32. All models are trained around 100 epochs and optimized by Adam optimizer [27]. The beam size is typically set to 10. When pretraining the diagram encoder, we first fill the diagram with a white background to make it equal in length and width, and resize it to 224×224. Then, we utilize the diagram feature extracted by the encoder to predict jigsaw location and geometry elements simultaneously and optimize the diagram encoder to obtain an informative diagram feature with a learning rate of $1e^{-5}$. Finally, the loss weight of the knowledge points classification task is set to 1, to promote the overall understanding of problems.

C.2 The Effect of Different Beam Size

In general, we set the beam size to 10 for testing. In this section, we explore the influence of different beam size. After the searched sequence program is executed, there will be three situations: right answer, wrong answer, and no result. As shown in Table. 4, BS, Acc, and NR represent beam size, accuracy, and no result, respectively. As the beam size is larger, we get higher accuracy and a lower proportion of no result. When beam size equals 1, the NGS-Auxiliary outperforms baselines significantly. Our model can achieve the highest accuracy of 64.5% when beam size is 100.

Method	BS	Acc(%)	NR(%)
Seq2Prog + Diagram	1	31.7	58.3
	10	52.6	20.4
	100	58.3	5.86
NGS-Auxiliary	1	45.6	42.6
	10	60.7	14.6
	100	64.5	2.95

Table 4: Performance comparison under different beam size settings.

C.3 Ablation Study

As shown in Fig. 3, we conduct experiments to evaluate the contribution of different auxiliary tasks. '+' represents we add the auxiliary component. NGS-Auxiliary means that adding all three auxiliary tasks together. We consider six different combinations: 1) only the NGS; 2) NGS + Geometry Elements (NGS+GE); 3) NGS + Jigsaw Location (NGS+JL); 4) NGS + Knowledge Points (NGS+KP); 5) NGS + diagram-based pretraining (NGS+JL+GE); 6) NGS with all three auxiliary tasks (NGS-Auxiliary). We can see that all three auxiliary tasks can promote the performance of NGS. The accuracy gains of GE, JL, KP, JL + GE, and combining all three tasks are 0.9%, 1.2%, 2.2%, 2.9%, 4.0%, respectively. These results show that all our

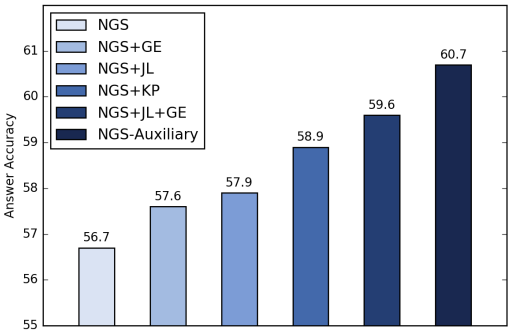

Figure 3: Ablation study on different auxiliary components.

self-supervised and auxiliary tasks can enhance the comprehensive understanding and multimodal reasoning ability of NGS.

C.4 Case Analysis

As shown in Fig. 4, we select two typical cases to demonstrate the programs generated by different models and some representative errors.

In the left case, the knowledge of the Pythagorean theorem, tangent, and square are required for solving the problem. Baseline method and our NGS fail to generate the correct operations and get no result. However, our NGS-Auxiliary successfully predicts the use of knowledge in the proposed auxiliary task. And more importantly, the correct "PythagoreanMinus" program is generated, and the right answer is obtained.

The right one is a typical error case, in which model needs to understand a complex scene. Although NGS-Auxiliary has predicted the knowledge points of Proportion program correctly, all three models fail to predict the correct answer. A better multimodal method is required to handle this hard high-level reasoning task in the future.

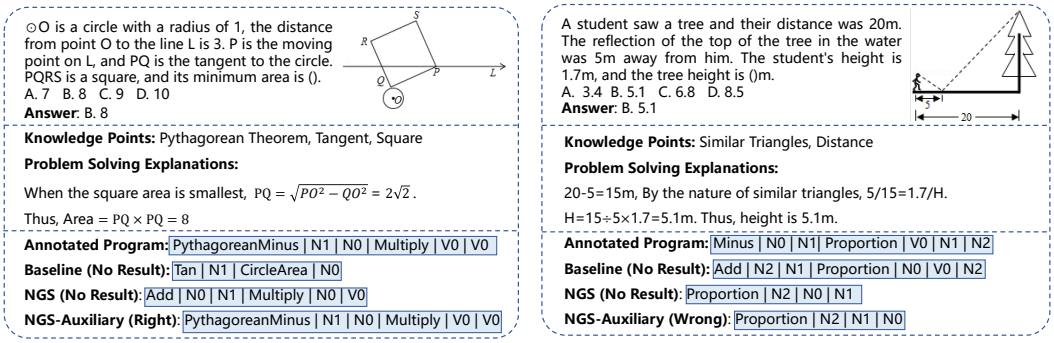

Figure 4: Typical cases. No Result represents the answer executed by the programs is not in the options, and Wrong represents getting a wrong option. Baseline is a “Seq2Prog + Diagram" model. In the case on the left, NGS-Auxiliary successfully predicts the knowledge of the Pythagorean theorem through auxiliary tasks and get the right answer. For the case on the right, the problem is quite hard that current model cannot solve it.

D Related Work

Geometry Problems Solving Developing automated systems to solve geometry problems has a long history in AI [28, 29, 30, 23]. For example, [9, 10] built the first automated system, GeoS, to solve SAT style geometry problems. GeoS used NLP and computer vision techniques (e.g., OCR) to parse a geometry problem’s text and diagram jointly as logic forms. However, this system highly depended on the manually designed logic forms and was only examined in a small dataset with 185 problems. Besides, the limited logic forms are hard to cover various geometry problems, leading to low generalization. To improve GeoS, [11, 25] replaced these handcraft constraints with geometry axiomatic knowledge in the form of horn-clause rules, but their dataset and code are not released. Recently, several new works [31, 32] also have emerged in the field of geometric problem solving.

Multimodal Reasoning Visual question answering is a representative multimodal task that requires the model to have reasoning ability [33, 18]. [34] built a new diagnostic VQA dataset (called CLEVR) with annotated functional programs. Based on this benchmark, some methods proposed an implicit reasoning framework to jointly encode multimodal information [16, 17]. Moreover, several works [35, 36] utilize domain-specific languages to perform explicit symbolic reasoning. However, these program languages only consider elementary operations, such as counting objects. They are not directly applicable to geometric problems, which require multiple steps of numerical calculation and involve theorem knowledge.

Self-supervised Auxiliary Task Self-supervised pretraining has gradually emerged [37, 38] as a effective technique to deal with label scarcity and improve model performance. To enhance visual features, most of these methods construct pseudo labels automatically and train on auxiliary tasks, including image jigsaw [39, 40], inpainting [41], super resolution [42], etc. Inspired by these works, we design two self-supervised auxiliary tasks and a supervised auxiliary task to enhance the reasoning ability of our NGS.