Advancing Mathematical Reasoning with Deep Learning: From Numerical Insights to Geometrical Understanding

Jiaxin Zhang

Computer and Information Sciences

University of Strathclyde

Thesis submitted for the degree of *Doctor of Philosophy*

June 2024

Glasgow

Date: 31st Mar 2024

Abstract

The field of automated mathematical reasoning has captured the interest of the AI community since the last century, acknowledged as a key step towards achieving true artificial intelligence. This research domain's evolution transits through rule-based approaches, semantic parsing, statistical machine learning, and recently, deep learning techniques. Moreover, automated mathematical reasoning has found extensive commercial applications. Educational enterprises have begun leveraging AI models for intelligent tutoring systems to assist students with mathematical problems. In the financial sector, it aids in analysing complex financial reports, with firms like JP Morgan incorporating AI to enhance their analysis capabilities.

This thesis concentrates on two distinct tasks within automated mathematical reasoning: text-based numerical reasoning and automated geometry maths problem-solving. Current methods face challenges in addressing complex mathematical reasoning tasks, evident in the lengthy and diverse solutions required. Additionally, in solving geometry maths problems, there is a noticeable deficiency in models' abilities to accurately interpret geometric relationships from diagrams, which compromises their effectiveness. Furthermore, the advent of large language models (LLMs) and multi-modal models (MMs) underscores the need for a standardised benchmark to evaluate these models' abilities in geometry problem-solving.

To address these issues, we introduce the ELASTIC model in this thesis, designed for text-based numerical reasoning task. ELASTIC uniquely separates the generation of operators and operands to minimise errors from complex reasoning chains and is versatile enough to accommodate a varying number of operands per operator. This makes it broadly applicable across different domains. Our experimental results show

ELASTIC's superior performance, significantly outperforming prior models.

Furthermore, we extend the application of the ELASTIC model to tackle geometry maths problems, which are inherently more complex due to the inclusion of geometric diagrams and a broader variety of problem types. To navigate these complexities, we propose the Geometry-Aware Problem Solver (GAPS), a model specifically crafted to solve diverse types of geometric maths problems by generating tailored solution programs. Our experiments validate GAPS's advancement over existing methods.

However, we observed that direct vector representation of geometric diagrams fails to capture the complex geometric relationships, which are critical in solving geometry maths problems. To overcome this, we propose converting geometric relationships into natural language, integrating them with the textual problem descriptions. This method not only improves the interpretability and effectiveness of the models but also allows for the utilisation of LLMs in generating reasoning programs.

Lastly, despite the impressive capabilities of recent LLMs and MMs, their proficiency in solving geometry problems, requiring an integrated understanding of textual and visual information, remains unexplored. To fill this gap, we introduce the GeoEval benchmark in this thesis. Through extensive evaluation with GeoEval, we provide a comprehensive quantitative evaluation of the latest LLMs and MMs in geometry problem-solving task. This research marks a significant step forward in assessing the capabilities of state-of-the-art AI models in the realm of geometry problem-solving task.

Contents

Abstract ii

List of Figures xii

List of Tables xix

Preface/Acknowledgements xxvii

I Introduction and Preliminary 2

1 Introduction 3

1.1 Introduction to the Automated Mathematical Reasoning 3

1.2 Motivation 6

1.2.1 Text-based Automated Numerical Reasoning 6

1.2.2 Automated Geometry Maths Problem-Solving 8

1.3 Thesis Statement 11

1.4 Research Objectives 12

1.5 Thesis Layout 14

1.6 Publications 15

2 Preliminaries and Related Work 16

2.1 Review on Large Language Models 16

2.1.1 Transformer Architecture 17

2.1.1.1 Encoder 18

2.1.1.2 Decoder 19

2.1.1.3 Multi-Head Attention 19

2.1.2 Types of Large Language Models 21

2.1.3 Pre-training Large Language Models 22

2.1.4 Task-specific Fine-tune Large Language Models 22

2.1.5 In-Context Learning 23

2.2 Text-based Automated Numerical Reasoning Overview 24

2.2.1 Benchmarks for Text-based Numerical Reasoning Task 24

2.2.1.1 Datasets of Maths Word Problems 24

2.2.1.2 Datasets for Maths Question Answering 25

2.2.2 Methods for Text-based Numerical Reasoning Task 26

2.2.2.1 Rule-based and Statistical Methods 27

2.2.2.2 Deep Learning Methods: Sequence-to-Sequence/Tree Methods 28

2.2.2.3 Large-Language-Models-based Methods 37

2.3 Automated Geometry Maths Problem-solving Overview 41

2.3.1 Benchmarks for Geometry Mathematical Problems 42

2.3.2 Methods for Geometry Maths Problem-solving 43

2.4 Chapter Summary 46

II Text-based Automated Numerical Reasoning 48

3 ELASTIC: Numerical Reasoning with Adaptive Symbolic Compiler 50

3.1 Introduction 51

3.1.1 Contributions 53

3.2 Approach 53

3.2.1 Task Definition 55

3.2.2 Encoder Part 56

3.2.3 Compiler Part 56

3.2.3.1 Decoding Vocabulary and Token Embedding 56

3.2.3.2 Reasoning Manager 56
3.2.3.3 Operator Generator 57
3.2.3.4 Operands Generator 58
3.2.3.5 Memory Register 58
3.2.4 Training Objective 59
3.3 Experimental Setup 59
3.3.1 Datasets 59
3.3.2 Baselines 60
3.3.3 Implementation Details 61
3.4 Experimental Results and Discussions 61
3.4.1 Comparison with Previous Methods (RQ-3.1, RQ-3.2, RQ-3.3) . 61
3.4.2 Performance on Different Program Steps (RQ-3.4) 63
3.4.3 Necessity of Memory Register (RQ-3.5) 65
3.5 Chapter Summary 66

III Automated Geometry Maths Problem-Solving **69**

4 GeoEval Benchmark **71**
4.1 Introduction 72
4.1.1 Contributions 73
4.2 GeoEval Dataset 73
4.2.1 Data Collection 74
4.2.1.1 Collection from Diverse Data Sources 74
4.2.1.2 Backward Data Generation 74
4.2.1.3 Augmented Data Generation 75
4.2.1.4 Hard Data Collection 75
4.2.2 Features of GeoEval 76
4.3 Chapter Summary 77

5 GAPS: Geometry-Aware Problem Solver **79**
5.1 Introduction 80

5.1.1 Contributions 83
5.2 Preliminary: VL-T5 84
5.3 Approach 85
5.3.1 Task Definition 86
5.3.1.1 Normalise the Representation of the Solution Program 87
5.3.1.2 Domain Specific Language 87
5.3.2 Joint Diagram and Problem Text Encoder 88
5.3.2.1 Ggeometric Elements Enhancement 89
5.3.3 Problem-Type Classifier 89
5.3.4 Geometry-Specific Program Generator 90
5.3.4.1 Generating Sub-Program Step-by-Step 90
5.3.4.2 Using one Program Generator for Different Problem Types 93
5.3.4.3 Updating the Cache Token 93
5.3.4.4 Hierarchical Beam Search for Operators and Operands 94
5.3.5 Training Objective 95
5.4 Experimental Setup 95
5.4.1 Datasets 96
5.4.2 Baselines 96
5.4.3 Evaluation Metrics 97
5.4.4 Implementation Details 97
5.5 Experimental Results and Discussions 98
5.5.1 Comparison with the State-of-the-art Methods (RQ-5.1) 98
5.5.1.1 Results when trained on both UniGeo CAL and PRV . 98
5.5.1.2 Results when solely trained on UniGeo CAL 99
5.5.1.3 Results when solely trained on UniGeo PRV 100
5.5.2 Comparison of performances according to the sub-program types (RQ-5.2) 100
5.5.3 Investigation for the Model's Weakness on Certain Problem Type (RQ-5.3) 100

5.5.3.1 Performance Comparison with Geoformer Using Top-1 and Top-10 Accuracy Metrics . . . 100
5.5.3.2 Analysing Prediction Consistency: Top-1 and Top-10 Accuracy Comparison . . . 102
5.5.3.3 Analysing Performance Discrepancy: Complexity of CAL and PRV Problems . . . 103
5.5.4 Dataset Augmentation with PGPS9K: A Comparative Analysis (RQ-5.4) . . . 104
5.5.5 Analysis of the Utility of the Problem-Type Classifier (RQ-5.5, RQ-5.6, RQ-5.7) . . . 106
5.5.5.1 The Impact of Problem-Type Classifier on GAPS Performance . . . 106
5.5.5.2 Evaluating the Influence of Fine-Grained Problem-Type Classification on GAPS Performance . . . 107
5.5.6 GAPS Performance on GeoEval Benchmark . . . 108
5.5.7 Analysis for the Different Cache Token Updating Methods (RQ-5.8) 110
5.5.8 Enhancing Solution Program Generation with Hierarchical Beam Search (RQ-5.9) . . . 111
5.5.9 Enhancing geometric element Selection with the geometric elements Enhancement Method (RQ-5.10) . . . 112
5.6 Chapter Summary . . . 113

6 GOLD: Geometry Problem Solvers with Natural Language Description 116
6.1 Introduction . . . 117
6.1.1 Contributions . . . 119
6.2 Approach . . . 120
6.2.1 Task Description and Pre-parsing . . . 120
6.2.2 Mapping Symbols and Geometric Primitives Separately . . . 121
6.2.2.1 Constructing the *feature_embedding* . . . 121
6.2.2.2 Constructing the *spatial_embedding* . . . 122

6.2.2.3 Symbol Vector and Geometric Primitive Vector 123

6.2.3 Relation Construction Head 123

6.2.3.1 sym2geo relation 124

6.2.3.2 geo2geo relation 125

6.2.4 Problem-Solving Module 126

6.2.5 Training Objective 126

6.2.6 Inference Stage 127

6.2.6.1 Predict sym2geo Relation 127

6.2.6.2 Predict geo2geo Relation 129

6.2.6.3 Generate Solution Program 130

6.3 Experimental Setup 130

6.3.1 Datasets 130

6.3.2 Evaluation Metrics 131

6.4 Results and Discussions 131

6.4.1 Comparison with State-of-the-art Models (RQ-6.1) 131

6.4.2 Ablation Study on Natural Language Description (RQ-6.2) . . . 133

6.4.3 GOLD's Performance on GeoEval Benchmark 134

6.4.4 Accuracy of the Extraction of geo2geo and sym2geo Relations (RQ-6.3) 137

6.4.5 Influence of feature_embedding and spatial_embedding on Geometry Problem Solving (RQ-6.4) 138

6.4.6 Usefulness of the Additional geo2geo Relation (RQ-6.5) 138

6.5 Chapter Summary 140

7 Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving 142

7.1 Introduction 143

7.1.1 Contributions 144

7.2 Evaluating LLMs and MMs on GeoEval benchmark 145

7.2.1 Experimental Setup 145

7.2.2 Evaluation Metric 146

7.2.3 Experimental Results (RQ-7.1) 146
7.2.3.1 Comparison among LLMs 147
7.2.3.2 Comparison among Multi-Modal Models 148
7.2.3.3 Comparison between LLMs and Multi-Modal Models . 148
7.2.3.4 Analysis on the Best Model (RQ-7.2) 149
7.2.4 Benefit from the Geometric Diagram Descriptions (RQ-7.3) . . . 150
7.2.5 External Knowledge Required (RQ-7.4) 150
7.2.6 Performances According to Different Problem Lengths and Varied Complexities (RQ-7.5, RQ-7.6) 151
7.3 Chapter Summary 153

IV Conclusion and Future Work 155

8 Conclusion and Future Work 156
8.1 Contributions 158
8.2 Findings and Conclusions 160
8.2.1 Limitations of State-of-The-Art Methods on Text-based Automated Numerical Reasoning Task 161
8.2.2 Generalise the Maths Reasoning Ability From Text-based Data to Image-text Data 163
8.2.3 Improve the Quality of Fusing the Diagram and the Text Modality for Solving Geometry Maths Problems 165
8.2.4 Evaluate Latest LLMs and MMs Proficiency in Geometry Maths Problem-Solving 166
8.3 Future Work 168
8.3.1 Other Multi-Modal Mathematical Reasoning 168
8.3.2 Different Formats to Represent the Reasoning Process 169
8.4 Chapter Summary 170

Bibliography 170

A ELASTIC: Numerical Reasoning with Adaptive Symbolic Compiler 217

A.1 Example for a Maths Word Problems . . . 217

A.2 An Example to Show How Separated Modules Work . . . 218

A.3 Comparison between Operation Length of the Golden Numerical Reasoning Program in MathQA and FinQA Datasets . . . 219

A.4 Training (RoBERTa-large) on Extended FinQA Dataset) . . . 219

B GeoEval Dataset 223

B.1 Source datasets for GeoEval-2000 dataset . . . 223

B.2 Inspection of GeoEval-hard subset . . . 225

B.3 Examples from GeoEval Representing Five Features . . . 225

B.3.1 Comprehensive Variety . . . 225

B.3.2 Varied Problems . . . 225

B.3.3 Dual Inputs . . . 227

B.3.4 Diverse Challenges . . . 227

B.3.5 Complexity Ratings . . . 227

B.4 Algorithm for Classifying Geometry Math Problems Complexity . . . 229

C GAPS: Geometry-Aware Problem Solver 230

C.1 Hierarchical Beam Search . . . 230

C.2 Accelerated Convergence with Problem-Type Classifier in GAPS Training 230

C.3 Case Study for the Geometric Element Enhancement Technology . . . 230

D GOLD: Geometry Problem Solvers with Natural Language Description 234

D.1 Preliminary: FCOS and GSM Models . . . 234

D.1.1 FCOS: Object Detection . . . 234

D.1.1.1 Bounding Box Regression . . . 235

D.1.1.2 Object Classification . . . 236

D.1.1.3 Centerness Prediction . . . 236

D.1.1.4 Train the FCOS Network . . . 237

D.1.2 GSM: Instance Segmentation . . . 237

D.2 Convert Relations to Natural Language Descriptions 238
D.3 Instruction Choice 239

E Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving 242

E.1 Model Hyperparameters 242
E.2 Instruction Prompt Used for Evaluating Models 244
E.3 Prompt and Heuristic rules For Answer Extraction 246
E.4 Results Across Different Subjects 247

List of Figures

1.1 Two typical geometry problems from the UniGeo dataset [1]. Particularly, the problems in the blue rectangle box belongs to the calculation problem, whereas the problems in the orange rectangle box belongs to the proving problem. The operand “C_x” refers to the x-th constant, the operand “N_x” refers to the x-th numbers in the problem text, “V_x” refers to the results of the previous sub-program that at the x-th index of the total program, and the operand “E_x” refers to the x-th geometric element from the problem text. The operator ”R_x” refers to the x-th pre-defined theorem. 9

2.1 The Transformer architecture, image taken from the [2]. 18

2.2 The left is the Scaled Dot-Product calculation, and the right is the Multi-Head version. The image is taken from [2]. 20

2.3 The Seq2Seq model for mathematical reasoning task. Here, **W** represents the textual words in maths problem texts and **N** signifies the numerical elements within the problem texts. The encoder converts each word in the text into a hidden state vector **h**. These vectors are subsequently utilised as the initial input alongside a start token **S** for the decoder. The decoder then generates an output at each step, which serves as the subsequent input for the following step in the decoding process. 30

2.4 The Seq2Seq model incorporating attention for mathematical reasoning task. During the decoding phase for step j, the decoder's hidden state h_j^{dec} engages in the computation of attention scores in relation to all the encoder's hidden states. These attention scores are then applied to perform a weighted summation of the encoder's hidden states, resulting in the context vector c_j. This context vector, along with the hidden state h_j^{dec}, contributes to the generation of the probability distribution across the decoding vocabulary. From this distribution, the word "N1" is selected for having the highest probability. 31
2.5 The Seq2Seq with Pointer-Network for mathematical reasoning task. During the decoding phase at time step j, the model computes the weight p_{gen} for the generation probability using the decoder's hidden state h_j^{dec} in conjunction with the context vector c_j. Subsequently, the decoding vocabulary's probability distribution is adjusted by adding the product of attention weights multiplied $(1 - p_{gen})$. The word "N1", which holds the maximum probability within the final distribution, is then chosen as this time step's output. 33
2.6 Different representations of the equation for a MWP. Prefix/Suffix refers to the pre/post order traverse of the AST. The figure is from [3]. 34

3.1 The overall architecture of the ELASTIC model. The Encoder part takes the sequence of question text Q and passage text P as input, then generates the contextual vectors $\mathbf{h}^{enc}$. The Compiler part consists of four modules: **Reasoning Manager**, **Operator Generator**, **Operands Generator**, and **Memory Register**. The right part of the figure shows a complete process of the generation of sub-program r_t. Firstly, Reasoning Manager sends the guidance vectors g^{op} to the Operator Generator, which guides the generation of operator op_t. Secondly, Reasoning Manager suspends the Operator Generator, then the Operands Generator takes g^{op} and op_t from the Operator Generator to produce the first operand oe^t_1. When finish the generation of the sub-program r_t, the Memory Register stores the results and updates the embedding vectors of cache token $\#t$ by g^{oe}_t. Again, the Compiler repeats to generate next sub-program r_{t+1}. 54

3.2 (a) Program Accuracy on FinQA according to the M-MDD. (b) Program Accuracy on MathQA according to the M-MDD. (c) Program Accuracy and Operator Accuracy of ELASTIC (RoBERTa-large) on different program steps in MathQA dataset, compared with Program Accuracy of FinQANet (RoBERTa-large). 65

4.1 Example for the backward question. 74

5.1 An illustration of the VL-T5 model. The image is taken from the original paper [4]. 84

5.2 First, the diagram is divided into patches, and along with the problem text, both are transformed into vectors. These vectors are then concatenated and passed to the joint diagram and problem encoder, producing the joint representations denoted as H. Subsequently, the geometry-specific program generator starts to generate the solution program, each solution program is represented by several sub-program r^i_{sub}. Each sub-program r^i_{sub} contains one operator op_i and corresponding operands $\{oe^i\}$, where the query vector q^{op} is used for generating the operator op^i, and the query vector q^{oe} is used for generating operands in the sub-program. After each sub-program is generated, the corresponding cache token is updated by replacing its vector by the q^{oe}_t, which is the query vector used for generating the last operand in the sub-program. Specifically, P_{type} is used to produce the mask to multiply with the probabilities of operators and operands, thus influencing GAPS to select op and oe from the domain-specific language (DSL) of the corresponding problem type. This mechanism enables GAPS to adapt and solve different kinds of geometry maths problems effectively. 86
5.3 An example of normalised representation of the solution program. . . . 88
5.4 Overall results comparison in top-10 accuracy between GAPS and baselines on all sub-tasks by the radar chart. 101
5.5 (a) Numbers of occurring indices of correct predictions in top-10 predictions candidates. We discard 2th, 3th, and 7th indices because numbers are zero. (b) Distributions of number of operands following each operator in ground truth solution programs from CAL problems and PRV problems. (c) The number of incorrect predictions due to wrong operators or wrong operands generated by GAPS in all sub-tasks. The “Angle”, “Length”, and “Other” belong to CAL problems, and the others belong to PRV problems. 102

6.1 The illustration of the GOLD Model. The diagram $\mathcal{D}$, problem text $\mathcal{T}$, and solution program $\mathcal{P}$ used in this illustration are sourced from the PGPS9K dataset [5]. The symbols and geometric primitives in the diagram are annotated using the notations from the Notation Table, which are consistent with the colours of extracted relations of sym2geo and geo2geo. 120

6.2 Top-left: the performance of the GOLD (using T5-base) with (**w**) and without (**w/o**) the *geo2geo*. Top-right: Geometry maths problem. Bottom: Predicted diagram description with and without the *geo2geo*. The same text between (**w**) and (**w/o**) is omitted for space consideration, where the red text is *geo2geo* relations. 139

7.1 Comparison of models requiring external knowledge ("w" in blue colour) and those do not ("w/o" in orange colour). 151

7.2 Models performances on GeoEval-2000 subset according to different question lengths. 152

7.3 Model performances on GeoEval-2000 subset according to different complexity levels. 152

A.1 An example of maths word problems, using nations defined in Table 3.1. 218

A.2 An Example to Show How Separated Modules Work. 219

A.3 The distribution of Operation Length in FinQA and MathQA datasets. 220

B.1 Distributions of source datasets. 224

B.2 Distributions of different geometric shapes. 224

B.3 Examples from GeoEval-2000 dataset. The golden answer choice is highlighted in red color. 226

B.4 Examples of the flat geometry problem, the analytic geometry problem, and the solid geometry problem in GeoEval benchmark. 227

B.5 Examples of GeoEval-2000, GeoEval-backward, GeoEval-aug, GeoEval-hard subsets. 228

B.6 Example for a problem annotated with complexity in the GeoEval benchmark. 228

C.1 Comparison of training loss convergence in GAPS, with and without the utilisation of the problem-type classifier. The "loss" and "loss_wo" are the sum of operator and operands training losses values with and without problem-type classifier. The "op_loss" and "op_loss_wo" are the operator training loss values with and without problem-type classifier. The "oe_loss" and "oe_loss_wo" are the operand training loss values with and without problem-type classifier. The loss values are scaled by the logarithmic function for better visualisation. 232

C.2 This heatmap illustrates the probability distributions across operand generation with and without the utilisation of geometric element enhancement technology. It exemplifies the creation of a solution program for a geometry proving problem. We truncate the x-axis by omitting the constants not pertinent to solving proving problems. The y-axis denotes the tokens in the solution program, and the probabilities of generating operators are nullified, as this instance aims to scrutinise the impact of geometric element enhancement on the generation of operands. With the enhancement enabled, the value vectors for the geometric elements are derived from the vectors of the corresponding appended geometric elements in the problem text. Conversely, without the enhancement, the value vectors for the geometric elements are extracted from the vectors of the original geometric elements within the problem text. 233

D.1 The illustration of the FCOS network. The C2, C3, C4, C5 refer to the feature maps from the backbone network, such as ResNet-101 [6] and MobileNets [7]. Each of them are obtained by using different ratio (denotes as $/s$ (s = 8, 16, 32, 64, 128) to down-sample over the input image with the size of 800×1024 in this example. The figure is taken from the FCOS paper [8]. 235

E.1 Detailed accuracy scores for models across various academic subjects. . 247

List of Tables

1.1 An Example (from MathQA [9] dataset) requires solving the problem by conducting mathematical reasoning. The reasoning process could be represented by different formats, such as (a) Formula Format, (b) Sequential Format, (c) Pre-order Traverse Format, (d) Flattened Format, (e) Nested Format. The #n refers to the executable result from the n-th sub-program, and const_2 refers to the constant number 2. 7

2.1 Table listing all introduced datasets for MWPs and MATH-QA tasks. . 27

2.2 Datasets for the automated geometry maths problem-solving. − indicates "not available". 43

3.1 Task Definition Notation. 55

3.2 Overall Results for the baselines and ELASTIC on the testing data from three datasets. † means that the scores are taken from the original papers. ‡ means that the scores are taken from the FinQA paper [10]. ⋆ The program accuracy does not apply to the NumNet on FinQA and MathQA datasets because NumNet does not generate the intermediate reasoning program. In addition, NumNet could only solve reasoning program involving add and subtract operations. However, the proportions of examples only use add and subtract as operations in MahtQA are 0.055% and 0.056%, respectively. As a result, we choose not to train NumNet on MathQA. 62

3.3 ELASTIC and FinQANet performances on FinQA dataset in terms of different program steps. The "# Train & Dev" is the number of training and development data. All models use the RoBERTa-large as the encoder. † means the results of that model are taken from the original FinQA [10] paper. ‡ means that the FinQANet (RoBERTa-large) is re-trained by ourselves. 64

3.4 The performances of ELASTIC with or without memory register (MR). ELASTIC with MR performs better than without MR on FinQA and MathQA datasets. Both ELASTIC with or without MR performs better than FinQANet. All models use the RoBERTa-large as the encoder. . . 65

4.1 Statistics of GeoEval benchmark. 73

4.2 Comparison between GeoEval-hard with other public datasets. 76

4.3 Comparison between GeoEval benchmark and other datasets. Under *Comprehensive Variety*, MathVista and GeoEval stand out as collective datasets, while the rest, denoted as "n/a". GeoEval includes problems from seven public datasets and three newly created ones. *Varied Problems* categorises problems into "flat geometry", "solid geometry", and "analytic geometry", For *Dual Inputs*, "text" signifies problems presented only in text format, whereas "text + diagram" encompasses problems with both texts and diagrams. In *Diverse Challenges*, the symbol ‡ indicates that MathVista introduces three new datasets, which, however, are unrelated to the geometry problem-solving task. 76

5.1 The notations used for the task definition. 87

5.2 The top-10 accuracy (%) comparison between baselines and our proposed GAPS model. ⋄, †, and ‡ refer to “No Solution Programs”, “Encode Text and Diagram Separately”, and “geometry-specific Models”, respectively. We report performances separately when training models on only CAL problems, only PRV problems, and the entire UniGeo problems (see column header “Data”). Numbers in bold refer to the highest accuracy in the specific data subset. In addition, we report the accuracy differences between GAPS and the previous state-of-the-art model (Geoformer) in the parenthesis. Except for the results of Seq2Prog are taken from [11], other baselines’ results are taken from [1]. 98

5.3 Comparison between top-1 and top-10 accuracy (%) on CAL problems and PRV problems achieved by Geoformer and GAPS, when training models on the entire UniGeo dataset. 101

5.4 The performance of top-10 accuracy (%) achieved by GAPS model on various datasets, inlcuding UniGeo CAL, UniGeo PRV, PGPS9K test, and PGPS9K test datasets. The column displaying the highest accuracy is marked in bold. For datasets that were not used during the training stage, the accuracy is not reported, and the corresponding cells are marked as “n/a.” 105

5.5 The comparison between the top-10 accuracy (%) achieved by GAPS model under three configurations: 1. without the problem-type classifier (denoted as “w/o”), 2. equipped with the problem-type classifier (denoted as “w”), 3. considering UniGeo CAL, UniGeo PRV, and PGPS9K as three different problem-types (denoted as “three types”). The highest accuracy scores in each row are highlighted in bold type, indicating the best performance achieved by the respective configuration for each problem type. 107

5.6 Top-10 accuracy (%) of the GAPS model on the GeoEval benchmark. The † denotes that GAPS was evaluated on partial subsets of GeoEval-2000 and GeoEval-aug, as it can only solve geometry math problems with available diagrams. Approximately 62% of problems in the GeoEval-2000 and GeoEval-aug subsets contain diagrams, sourced from the PGPS9K, UniGeo, GeoQA+, and Geometry3K datasets. 108

5.7 Performance comparison between different cache token updating methods. q_0^{op} refers to the query vector used for generating the operator for the sub-program, e_{op} refers to the embedding of the generated operator for the sub-program, and q_t^{oe} refers to the query vector used for generating the last operand for the sub-program. For an equitable evaluation, GAPS models trained exclusively on UniGeo CAL, UniGeo PRV, and PGPS9K datasets are used to assess the performance on their respective test datasets. 110

5.8 Comparing Accuracy (%) of GAPS with Hierarchical Beam Search and Greedy Decode for Solution Program Generation. For an equitable evaluation, GAPS models trained exclusively on UniGeo CAL, UniGeo PRV, and PGPS9K datasets are used to assess the performance on their respective test datasets. 111

5.9 Performance comparison between GAPS with (w) geometric elements enhancement and without (w/o) it. 112

6.1 Comparison results on the test subsets of chosen datasets. PGPSNet reported models' performances using the ground truth diagram annotations, where these models have "(GT)" behind them. We re-implemented these methods to get performances without GT annotations. Note that UniGeo lacks GT diagram annotations, so relevant cells are "n/a". "†" and "‡" indicates the results are from [1] and [5], respectively. The performance of GAPS on the PGPS9K Test and Geometry 3K Test datasets, indicated by "◊" is evaluated using top-3 accuracy, the same metric employed by other models. 132

6.2 Evaluation of the GOLD model on two datasets with no description (n/a), logic-forms (LF), and natural language descriptions (NLD). Both the mean and standard errors of the accuracy metrics are presented. . . 134

6.3 The top-10 accuracy (%) of GOLD model achieved on GeoEval benchmark. The † denotes that GOLD was evaluated on partial subsets of GeoEval-2000 and GeoEval-aug, as it can only solve geometry math problems with available diagrams. Approximately 62% of problems in the GeoEval-2000 and GeoEval-aug subsets contain diagrams, sourced from the PGPS9K, UniGeo, GeoQA+, and Geometry3K datasets. The ‡ symbol indicates that GOLD utilizes only the problem text from the GeoEval-backward subset as input, since no corresponding geometric diagrams are available for that subset. 135

6.4 The check mark (✓) indicates that the corresponding embedding is enabled. Note that "pre-parsing" is not influenced by emb$_{feat}$ and emb$_{spat}$. Both the mean and standard errors of the accuracy metrics are presented. 138

6.5 Program accuracy with or without *feature_embedding* and *spatial_embedding*. The check mark (✓) indicates that the corresponding embedding is enabled. T5-base is used as the problem-solving module for the GOLD model. Both the mean and standard errors of the accuracy metrics are presented. 139

7.1 Accuracy scores of models on our GeoEval benchmark. The "◊" refers to all LLMs. The "A" signifies the overall accuracy across all problems, while "T" denotes the accuracy for problems containing only texts without diagrams. "n/a" indicates that scores are unavailable due to models cannot process text-only inputs. The "‡" notes that the accuracy figures for GPT-4V are derived from GPT-4, as GPT-4V does not support image-free inputs. 146

7.2 Comparison of models with (✓) and without (✗) geometric diagram descriptions. 150

A.1 The performances of ELASTIC (RoBERTa-large) on the test data from the combination of the original FinQA dataset and extended FinQA dataset, and only on the test data from the extended FinQA dataset. Note that the model is trained on the train data from the combination of the original FinQA dataset and the extended FinQA dataset. 221

A.2 An example from the extended FinQA dataset. The "Prediction" refers to the generated numerical reasoning program from ELASTIC (RoBERTa-large). †: The passage is from the FinQA dataset, which is reorganised for better readability. 221

B.1 The information of source datasets for GeoEval-2000 dataset. The "†" symbol indicates that proving problems from the UniGeo dataset have been excluded. The "‡" sign specifies that the count only pertains to geometry problems within the dataset, focusing on problems directly relevant to the GeoEval-2000's scope. 223

D.1 The defined paradigm used to convert *geo2geo* and *sym2geo* relations to natural language descriptions $\mathcal{L}$. "\${}" is the placeholder. The placeholder is filled in as demonstrated in the "Example" column, and the filled content is highlighted in bold type. 238

D.2 Two instruction templates. "IT" and "EX" refer to the instruction template and examples. The template in the left column is modified from the instruction used to train the Llama2 model, and another one is from the [12]. In the column of "IT", the "{problem Text}" is the geometry maths problem text $\mathcal{T}$, and "{Natural Language Descriptions}" is the description of the diagram $\mathcal{L}$. 241

E.1 The hyperparameters for the models used in the evaluation are detailed. When the "comments" section includes the format *model = ""*, it signifies that the model was loaded from the transformer package. The vLLM package indicates that models are implemented by the vLLM package, where more details can be found in `https://github.com/vllm-project/vllm`. For models other than OpenAI's GPT, custom codes were utilized for evaluation unless specified otherwise in the comments. . . . 243

E.2 Templates and examples provided illustrate the process of merging and instruction creation. The placeholder "${Merge}" represents the combined texts of "diagram descriptions," "problems texts", and "choice list". In cases where "diagram descriptions" are absent, the phrase "Here are the basic description of the diagram:" is omitted. 245

E.3 Task specific instruction used for extracting the answer, and three examples. 246

E.4 Regular expressions used for extracting the answers that GPT-4 cannot tackle. 247

Preface/Acknowledgements

"Don't Eat Fortune's Cookie", draws from Michael Lewis's speech in 2012, a constant reminder that the accomplishments of my PhD journey stem not solely from my endeavours but from a confluence of myriad supportive elements. The subject of this thesis, the automated solving mathematical reasoning problems, aims not only to aid human tasks but primarily to foster the advancement of intelligent education. I am convinced that the integration of AI technologies in education can empower individuals worldwide to access desired learning opportunities, thereby enabling them to pursue their passions in their fields of interest.

Foremost, I extend my heartfelt gratitude to my university for the generous scholarship that sustained my PhD journey, anchoring my commitment to my research. I am profoundly thankful to my supervisor, Dr. Yashar Moshfeghi, whose guidance transcended academic advice, offering patience and wisdom to navigate the challenges encountered along this journey. My colleagues at the NeuraSearch laboratory deserve special mention for their infectious enthusiasm and warmth, making my time in the UK feel like a second home.

I am indebted to my circle of friends - Jiaqi Xue, PengCheng Jing, ShuaiChen Chang, Sam, Jennie, Francesco, Zuzana, and Kunjira. Their companionship through the daunting and obscure corridors of PhD research has been an invaluable source of strength and cherished memories.

This thesis is dedicated to my wife, Yanjun Cheng, whose unwavering support and selfless sacrifice have been the bedrock of my pursuit. Her care for my well-being has been the light guiding me through the rigours of this journey. Yanjun, you are an integral part of my life's tapestry, interwoven with joyous memories, laughter, and emotional moments. My dear, you are my endless numeric expression of love, 100001000.

Lastly, but most importantly, I owe a profound debt of gratitude to my parents. Their boundless love and sacrifice laid the foundation for my dreams, making it possible for me to study abroad. Words fall short of expressing my appreciation for their unconditional love and care throughout my upbringing. The debt of love I owe to them is immeasurable, a testament to the sacrifices they've made to see me flourish. In the same vein of heartfelt appreciation, I extend my deepest gratitude towards my in-laws, affectionately known as Yan Ba and Yan Ma. Your unwavering support and love have been pillars of strength, allowing me to dedicate myself fully to my research without the burden of additional worries. Your generosity and kindness have been a beacon of light during challenging times, and I am profoundly grateful for the sense of peace and stability you've provided.

Additionally, I must express my profound gratitude to my grandmothers and my aunt, whose love and wisdom have been a constant source of comfort and guidance. Their unconditional support has been a foundation upon which I've built my aspirations and achievements. To my brothers and my brother's wife, who have upheld the fort at home with unparalleled dedication and care, your role in my life is immeasurable. The pleasure and gifts of your love and support are among the most cherished aspects of my journey, offering me a haven of peace and encouragement.

Each of you, in your unique ways, has contributed to the tapestry of my life and this academic endeavour. Your sacrifices, love, and encouragement have been the greatest gifts, enriching my PhD journey beyond measure. For this, I am eternally grateful and forever indebted.

Part I

Introduction and Preliminary

Chapter 1

Introduction

This thesis aims to investigate and advance the application of deep learning methods in solving complex mathematical reasoning problems, with a particular focus on text-based automated numerical reasoning task and automated geometry maths problem-solving task. This part includes Chapter 1 for introduction for the automated mathematical reasoning and Chapter 2 for introducing necessary preliminaries and reviewing relevant literature. In Chapter 1, we described the background of automated mathematical reasoning and then provided the motivation regarding to text-based automated numerical reasoning task and automated geometry maths problem-solving task, along with thesis statement, and research questions. In Chapter 2, we provided the preliminary and literature review of existing work, including methods for automated text-based numerical reasoning task and automated geometry maths problem-solving. We also described the benchmarks used for evaluating Artificial Intelligence (AI) methods' proficiency in solving the mathematical reasoning tasks.

1.1 Introduction to the Automated Mathematical Reasoning

The Turing test, which is based on whether a person can tell if they are chatting with an AI system or another human, was once seen as a key path towards creating truly intelligent machines [13]. But as AI rapidly evolved, many found this test wasn't the best way to measure an AI system's smart [14]. Today, researchers believe that if we want to measure how intelligent an AI system is, we should test it like we do with people. In fact, the human standardised maths test that we use to assess human intelligence is a better fit for evaluating the intelligence of the AI system than the Turning test [15, 16].

The human standardised maths assessment evaluates core elements of human intellectual capacity such as basic arithmetic, percentages, ratios, data analysis, and other intricate maths challenges [17]. This test measures mathematical reasoning, a vital cognitive skill that we use in different ways in our daily lives. From simple tasks like counting items to more complex ones like analysing financial tables or tackling maths word problems [18, 19]. Formally, mathematical reasoning means understanding numeric data presented in text, tables, or visuals and applying mathematical concepts to solve problems or draw conclusions [20]. For instance, a practical maths word problems of the primary school level, "*If a shirt is priced at £40 with a 25% discount, what's the final price?*", with the answer being "*£30*", relies on the foundational principles of mathematical reasoning. A more intricate example, such as determining the sales of a company, like: "*Given a 15% sales growth from 2018 to 2019 and 2018 sales of 10 thousands pounds, what were the 2019 sales?*", with the solution being "*£11,500*". To obtain the correct answer, one must select correct quantitative information, convert the unit, and execute detailed quantitative analyses.

Despite achieving levels of performance similar to human abilities in tasks necessitating cognitive recognition, the realm of AI still struggles with effectively matching humans in mathematical reasoning. Humans possess the capability to articulate their thought processes, such as formulating equations, to navigate through complex mathematical challenges. AI systems, however, due to their deficiency in mathematical reasoning, struggle to perform these precise and complex deterministic reasoning steps, which hinders their ability to derive conclusions. Nevertheless, for AI to be a useful aid in fields like education, finance, and physics, etc., it must contain robust mathematical reasoning skills. Consequently, there is an increasing focus on empowering AI systems with this critical ability, a domain that has intrigued researchers for decades.

Although the field of AI has reached human-like levels of performance in tasks that require cognitive recognition abilities, it continues to face challenges in executing mathematical reasoning on par with humans. Humans can naturally write down reasoning processes, such as equations, to work through complex maths problems. However, lacking mathematical reasoning ability, AI systems face challenges in conducting the accurate and intricate these deterministic reasoning steps, hindering them to conclude the answer. Yet, if we hope for AI to assist human beings in areas like education, finance, physics, and beyond, it must possess strong mathematical reasoning capabilities. As a result, there's a growing emphasis on equipping the next wave of AI with this skill, a topic that has caught researchers' attention for decades [21–25].

The ability of an AI system to perform mathematical reasoning can be evaluated through a variety of tasks, such as text-based automated numerical reasoning task [17, 18, 26–29]. In this scenario, the AI leverages its mathematical knowledge to analyse information and generate a solution. A specific example of this task involves the automated resolution of maths word problems (MWPs) [30–36]. MWPs generally require only basic mathematical knowledge, such as elementary arithmetic. When addressing an MWP, the AI must grasp the narrative context, identify numerical data and their interrelations, and then formulate a solution, like deriving an equation. For success in this area, the AI must exhibit skills in reading comprehension, semantic parsing, and mathematical logic. Another related example is answering mathematical questions, which, similar to MWPs, demands a broader spectrum of knowledge, such as tackling mathematics problems based on financial reports [10], science [37–41], and coding [42] domains. Another task, which differs from the previous that focus on textual data, is automated geometry maths problem-solving [43–47]. This task is more complex as it involves not just text but also geometric diagrams. To succeed in resolving the problem, not only does the AI system have to understand the problem text, but also to identify attributes and relationships from the diagram before producing a solution. This makes it a more challenging compared to tasks of solving MWPs and mathematical question answering. Apart from these tasks that focus on mathematical calculation problems, there's also automated theorem proving [48–56]. This task requires the AI system to produce logical sequences to prove mathematical theorems.

In this thesis, we aim to investigate and advance the application of deep learning methods in solving complex mathematical reasoning tasks, with a particular focus on numerical reasoning and geometry problem-solving. Additionally, we target to investigate the factors affecting the AI system's mathematical reasoning ability in tasks like MWPs, mathematical question answering, geometry problem-solving. Finally, this thesis proceed the evaluation of the current large language models (LLMs) and multi-modal models (MMs) proficiency in solving geometry maths problems, along with a new curated benchmark. The rest of this chapter introduces the motivations, statements and research questions of the thesis, and finally presents the outline of the remainder of this thesis.

1.2 Motivation

Section 1.2.1 presents an overview of the task of text-based automated numerical reasoning, encompassing solving of maths word problems (MWPs) and mathematics question answering (MATH-QA). Section 1.2.2 elaborates on the rationale behind choosing geometry problem-solving as a focus for further investigation, highlighting the inclusion of an additional modality, i.e., geometric diagrams. This section also explores the limitations of previous approaches that motivated our deep dive into this area of research.

1.2.1 Text-based Automated Numerical Reasoning

Automated solving MWPs and mathematical question answering (MATH-QA) require AI systems to comprehend text and identify relationships among the entities mentioned. Once these relationships are grasped, the AI system must then use its mathematical reasoning abilities to derive the correct solution. For such tasks, the input is typically a text that sets the context and poses a question. The output, on the other hand, is a sequence that depicts the reasoning process, composed of basic arithmetic actions (like addition or subtraction), discrete functions (such as comparison or counting), and numbers. An illustrative example of this task can be found in Table 1.1. The table presents a problem narrative followed by various formats that lay out the reasoning process. These formats combine both operators (e.g., arithmetic functions, discrete functions) and operands (e.g., numbers).

Table 1.1: An Example (from MathQA [9] dataset) requires solving the problem by conducting mathematical reasoning. The reasoning process could be represented by different formats, such as (a) Formula Format, (b) Sequential Format, (c) Pre-order Traverse Format, (d) Flattened Format, (e) Nested Format. The $\#n$ refers to the executable result from the n-th sub-program, and const_2 refers to the constant number 2.

Problem: A small table has a length of 12 inches and a breadth of b inches. Cubes are placed on the surface of the table so as to cover the entire surface. The maximum side of such cubes is found to be 4 inches. Also, a few such tables are arranged to form a square. The minimum length of side possible for such a square is 48 inches. What is the number for b?
(a) Formula Format: $\frac{48\times4}{12}$
(b) Sequential Format: $48 \times 4 \div 12$
(c) Pre-order Traverse Format: $\div, \times, 48, 4, 12$
(d) Flattened Format: multiply(48, 4)—(#0, 12)
(e) Nested Format: divide(multiply(48,4), 12)

Previous methods face two main limitations. First, they fail in complicated problems [57]. The complexity of such issues often arises due to the length of the reasoning program and the increased variety of operators employed. As the reasoning program becomes more extensive or the types of operators increase, the likelihood of errors, especially cascading ones, also rise. Second, previous models have adaptability issues. These stem from their inherent design or the way they represent reason processes, restricting their application across various fields. When tested on different domain datasets, many models fail because they lack the specific operators required for that domain. For instance, the NeRd model [58], optimised for the general DROP dataset [59], struggles on the FinQA [10], a finance-domain dataset.

So far, text-based automated numerical reasoning task still faces two primary challenges:

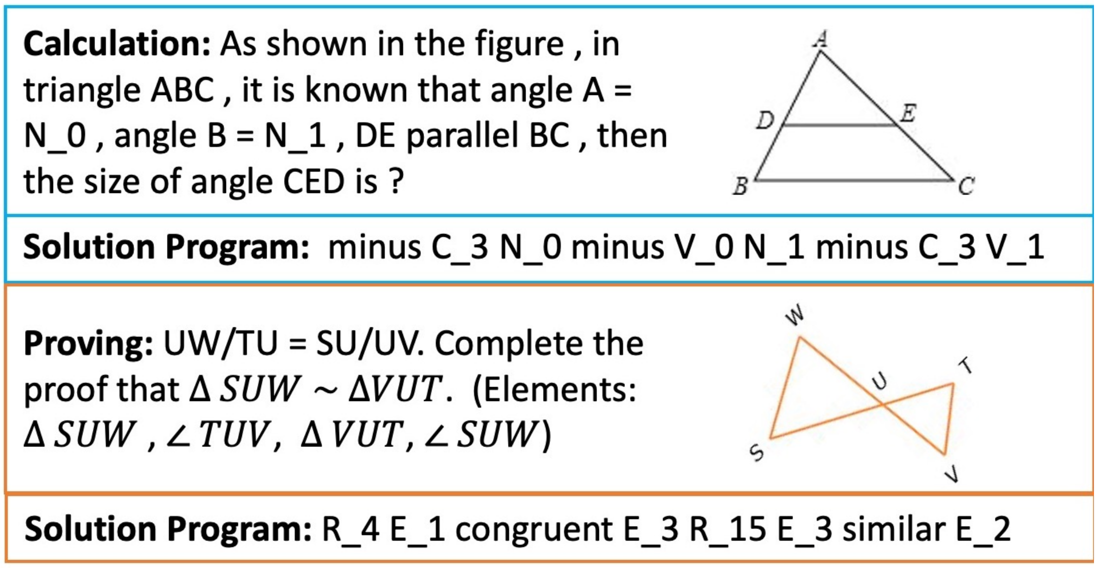
1. **Textual Understanding**: To excel in this task, AI system must possess strong semantic parsing and reading comprehension skills. This means identifying key entities and associated numerical details within the text. Furthermore, some questions might rely on implicit data not present in the text but are inferred from domain-specific knowledge, adding another layer of complexity to understanding the narrative.

2. **Precise Mathematical Reasoning**: The AI system must conduct the precise reasoning process. Even a single error can result in a domino effect of mistakes. However, many leading models tend to learn statistical patterns from the data rather than true mathematical reasoning. [31].

In this thesis, we aim to tackle these difficulties. We argue that by adopting a novel architecture for generating the reasoning process, we can enhance mathematical reasoning abilities. Such an architecture should address cascading errors arising from complex reasoning processes and offer easy extensibility. In Part II, we introduce a model tailored for this task. This model is designed to minimise the impact of cascading errors, especially in intricate reasoning scenarios. Moreover, its adaptability to various operators and operands counts following operators, allowing for seamless integration of external domain-knowledge.

1.2.2 Automated Geometry Maths Problem-Solving

As mentioned in Section 1.2.1, deep-learning methods has made impressive advancements in text-based automated numerical reasoning task. However, when it comes to multi-modal data, the research is still in the infancy. A notable example is the task of automated geometry maths problem-solving. While there have been efforts to adopt models successful with text-based numerical reasoning task for geometry problems, the results are not satisfactory. This challenge primarily arises because geometry problems, unlike other textual problems, come with the added complexity of diagrams. As depicted in Figure 1.1, not only is there a textual description and solution program, but also a diagram with geometric components and symbols. The objective is to synthesise this dual-modal information, both text and diagram, to effectively address the problem.



Calculation: As shown in the figure , in triangle ABC , it is known that angle A = N_0 , angle B = N_1 , DE parallel BC , then the size of angle CED is ?

Solution Program: minus C_3 N_0 minus V_0 N_1 minus C_3 V_1

Proving: UW/TU = SU/UV. Complete the proof that $\Delta SUW \sim \Delta VUT$. (Elements: ΔSUW , $\angle TUV$, ΔVUT, $\angle SUW$)

Solution Program: R_4 E_1 congruent E_3 R_15 E_3 similar E_2

Figure 1.1: Two typical geometry problems from the UniGeo dataset [1]. Particularly, the problems in the blue rectangle box belongs to the calculation problem, whereas the problems in the orange rectangle box belongs to the proving problem. The operand "C_x" refers to the x-th constant, the operand "N_x" refers to the x-th numbers in the problem text, "V_x" refers to the results of the previous sub-program that at the x-th index of the total program, and the operand "E_x" refers to the x-th geometric element from the problem text. The operator "R_x" refers to the x-th pre-defined theorem.

Recently, with the introduction of various public datasets, several neural-based methods have been proposed to tackle geometry problems, such as Inter-GPS [60], NGS [11], and DPE-NGS [61]. However, these methods often fall short in seamlessly integrating multiple modalities. Also, there's a prevalent focus on one kind of geometry problem, calculation, while other kinds like proving, are largely overlooked. Yet, it's imperative to design solutions catering to a broader range of types of geometry problems. Also, past research have encountered difficulties in accurately parsing geometric diagrams, especially when dealing with intricate spatial relationships or overlapping geometric elements [62]. This limited comprehension adversely affects problem-solving performance. While rule-based techniques might offer better diagram parsing, their rigid rules compromise adaptability and hinder integrating other powerful models.

Nowadays, automated geometry maths problem-solving still faces two major challenges:

1. **Parse the Diagram Precisely**: Geometry problem solving is a challenging task for deep-learning methods due to that it demands capabilities like parsing multi-modal data, executing symbolic abstraction, accessing theorem knowledge, and engaging in precise mathematical reasoning.
2. **Accompany Text with Diagram**: Beyond effectively parsing diagrams, any solution approach must also harmonise the information from both the diagram and the associated text, especially when it comes to geometric elements specified in the narrative.
3. **Assessing Advanced Models' Geometry Problem-Solving Ability**: Despite the emergence of large language models and multi-modal models, it remains uncertain how effectively these advanced models can automatically solve geometry maths problems. This uncertainty is primarily due to the absence of a standardised benchmark.

In this thesis, we target to resolve the above challenges. We posit that a unified deep-learning model, trained across diverse geometry problem types, can enhance mathematical reasoning capabilities. Moreover, we contend that merely translating geometric diagrams into high-dimensional vectors fails to capture their inherent complexity. As such, in Part III, we delve into advanced representation techniques for geometric diagrams, aiming for more precise and interpretable representations. In addition, we investigate for a effective modality-fusion method to cohesively understand different modalities, especially texts and diagrams.

1.3 Thesis Statement

This thesis states that the enhancement of text-based numerical reasoning and geometry math problem-solving capabilities can be achieved through the innovative architecture of the solution program generator and the straightforward representational approaches of the diagrams. Specifically, this thesis states that a new architecture, which independently generates operators and operands in reasoning programs, will minimize cascading errors and enhance the complexity and adaptability of reasoning programs. Additionally, the use of a single solver for various types of geometry problems will improve performance. This thesis also states that converting geometric diagrams into natural language descriptions, rather than high-dimensional vectors, will better capture intricate details and enhance the AI's reasoning abilities. Finally, this thesis states that the underperformance of current Large Language Models (LLMs) and Multimodal Models (MMs) on geometry problem-solving tasks highlights the need for a robust and varied benchmark to accurately assess their capabilities.

The detailed statements of the thesis are shows as follows:

- **Statement (1)**: For enhanced mathematical reasoning capabilities, we need a new decoder architecture that minimises the risk of cascading errors. Specifically, by independently generating the operators and operands in reasoning programs, the system can produce more complex reasoning programs. Additionally, this separation approach ensures the model isn't limited to a specific domain, enabling it to integrate a variety of operators and thereby enhancing its adaptability. (Chapter 3).

- **Statement (2)**: Using one solver for various geometry maths problems can enhance performance. Specifically, we believe that reasoning programs used to solve geometry maths problems share a common structure, comprising operators (like arithmetic functions and geometric theorems) and operands (such as numbers and geometric elements). Leveraging the architecture introduced in Chapter 3, we can efficiently harness the advantages presented by these new geometry problem types without any negative implications. (Chapter 5).

- **Statement (3)**: Previous deep-learning methods often convert geometry problem diagrams into high-dimensional vectors, which can negatively impact the system's ability to reason mathematically about these problems. The intricate details and overlaps commonly found in geometric diagrams require a new representation method that high-dimensional vectors cannot capture. As a result, using natural language descriptions for geometric diagrams becomes advantageous. These descriptions not only provide an accurate and understandable representation of the diagrams but also bridge the gap between textual descriptions (geometry problems) and visual representations (geometric diagrams). Adopting this method can improve the AI system's capability in tackling geometry maths problems. (Chapter 6).

- **Statement (4)**: As Large Language Models (LLMs) and Multimodal Models (MMs) advance, they demonstrate promising capabilities across a variety of tasks. However, while some LLMs and MMs show impressive results on certain mathematical reasoning benchmarks, we suspect these achievements might stem from issues of data leakage. Particularly in the domain of geometry problem-solving, LLMs and MMs tend to underperform. This underscores the need for a robust and varied benchmark specifically designed to assess the ability of LLMs and MMs in tackling geometry maths problems. (Chapter 4 and Chapter 7).

1.4 Research Objectives

This thesis primarily focuses on enhancing the AI system's mathematical reasoning capabilities for two tasks, text-based numerical reasoning and geometry maths problem-solving. The objective is to introduce new architectural frameworks tailored to address complex reasoning challenges and to delve into the pivotal components that boost mathematical reasoning proficiency. Specifically, the objectives are to:

(1) Examine the effect of segregating the generation of operators and operands on the mathematical reasoning skills for complex problems. Specifically, carry out both

quantitative and qualitative evaluations to determine how this innovative architecture surpasses current state-of-the-art methods across various public datasets.

(2) Following (1), delve into the methods to seamlessly integrate a range of operators, ensuring their efficiency isn't compromised by the number of operands they encompass.

(3) Building on (1), assess the model's resilience in handling complex reasoning tasks, especially as indicated by the number of steps in the program. Also, identify the challenges faced during the execution of these tasks.

(4) Examine the influence of mathematical reasoning for geometry problems when the model is trained on augmented datasets that encompass diverse problem types.

(5) Building on (4), delve into the mathematical reasoning capabilities across various types of geometry maths problems. Determine the underlying reasons for why the model performs significantly better on certain problem types compared to others.

(6) Propose a technique to depict geometric diagrams using natural language descriptions. Furthermore, evaluate how this natural language representation stands against the formal language employed by symbolic solvers concerning mathematical reasoning in geometry maths problems.

(7) Expanding on (6), study the degree to which using natural language descriptions for geometric diagrams can enhance the synergy between textual data and the diagrams. Furthermore, explore methods to integrate these natural language descriptions with existing language models.

(8) Building on findings from (4), (5), (6), and (7), this research delves into the capabilities of the most recent advancements in Large Language Models (LLMs) and multi-modal Models (MMs) in addressing geometry maths problems. Specifically, it aims to identify the critical elements that shape these models' effectiveness in geometry problem-solving tasks.

1.5 Thesis Layout

This thesis is divided into four parts and seven chapters:

- **PART I: Introduction and Preliminary**: This part contains Chapter 1 and Chapter 2, which provide the background, preliminary and related work of this thesis. Specifically, Chapter 2 begins with a concise overview of large language models. It then moves on to introduce the foundational understanding of text-based automated numerical reasoning and highlights prior research in this area. It concludes by discussing the evolution of automated geometry maths problem-solving.

- **PART II: Text-based Automated Numerical Reasoning**: This part contains Chapter 3. Within this chapter, a deep-learning model is introduced which segregates the generations of operators and operands. This distinction helps the model to be less susceptible to cascading errors arising from complicated reasoning. The specific contributions and comprehensive experiments related to this are thoroughly discussed in the respective chapters.

- **PART III: Automated Geometry Maths Problem-Solving**: This part comprises Chapter 4, Chapter 5, Chapter 6, and Chapter 7. Chapter 4 introduces the GeoEval benchmark, a comprehensive collection that focuses on automated geometry maths problem-solving task. Next, Building on the model introduced in Chapter 3, Chapter 5 addresses challenges in automated geometry maths problem solving by introducing a model tailored to tackle various types of geometry maths problems concurrently. Chapter 6 then enhances the mathematical reasoning capabilities for geometry problems by employing natural language descriptions for geometric diagrams. This approach facilitates a smooth integration with large language models. Collectively, this section underscores the potential of integrating multi-modal information to bolster automated mathematical reasoning. Finally, Chapter 7 facilitates a deeper investigation into the performance of LLMs and MMs on GeoEval benchmark.

- **PART IV Conclusion**: This part includes Chapter 8. Chapter 8 outlines the main contributions and discoveries of this thesis. Additionally, this chapter presents potential avenues for future research and developments.

1.6 Publications

The thesis is built from the following publications:

1. Jiaxin Zhang, Yashar Moshfeghi. ELASTIC: numerical reasoning with adaptive symbolic compile. Full paper accepted in 36th Advances in Neural Information Processing Systems. (**NeurIPS 2022**) (Chapter 3)

2. Jiaxin Zhang, Yinghui Jiang, Yashar Moshfeghi. GAPS: Geometry-Aware Problem Solver. The journal of Artificial Intelligence, Elsevier. (**AIJ**) (under-review) (Chapter 5)

3. Jiaxin Zhang, Yashar Moshfeghi. GOLD: Geometry Problem Solvers with Natural Language Description. Full paper accepted in 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics. (**NAACL 2024 Findings**) (Chapter 6)

4. Jiaxin Zhang, Zhongzhi Li, Mingliang Zhang, Fei Yin, Chenglin Liu, Yashar Moshfeghi. GeoEval: Benchmark for Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving. in 62nd Annual Meeting of the Association for Computational Linguistics. (**ACL 2024 Findings**) (Chapter 4 and Chapter 7)

Chapter 2

Preliminaries and Related Work

In this chapter, we lay out the foundational concepts underpinning this thesis. We start with an introduction to large language models (LLMs) in Section 2.1. Here, we trace the evolution of LLMs and explain essential technologies crucial for comprehending this thesis. Next, a comprehensive literature review on benchmarks and methods for text-based automatic numerical reasoning is presented in Section 2.2. Finally, We wrap up with Section 2.3, providing an overview of automated geometry maths problem-solving, encompassing both benchmarks and methodologies.

2.1 Review on Large Language Models

As highlighted in Section 1.1, for successful mathematical reasoning, an AI system needs to comprehend the problem text, a capability often termed as reading comprehension ability. Hence, there's a need for a model proficient in capturing the nuances from textual information. Recent large language models (LLMs) have proven adept at offering dense representations of data, thereby enhancing performance across various application domains, including computer vision [63–65], speech [66–68], and multi-modal [4,69–72].

Compared to other models, there are several reasons why LLMs have dominated in the research community, particularly when it comes to data representation:

1. **No Need for Annotated Data**: LLMs are typically pre-trained on unlabelled data using unsupervised learning methods. This is advantageous because there's

a vast amount of textual data accessible online that LLMs can utilise to grasp a general understanding of language. Furthermore, once pre-trained, LLMs can be fine-tuned with a relatively small set of labelled data to achieve top-tier results in numerous downstream applications.

2. **No Gradient Vanishing Problem**: Thanks to the self-attention mechanism [2] (detailed in the upcoming Section 2.1.1.3) and residual connection mechanism [6], LLMs can handle lengthy input data without facing the gradient vanishing issue. This allows them to adeptly capture long-range dependencies within the data.

3. **Parallel Processing**: LLMs can process all parts of a sequence simultaneously, which is more computationally efficient and allows for faster training with modern GPUs.

4. **Scalability**: LLMs can be scaled up easily according to the scaling laws [73]. Models like GPT-3 [74], llama-2 [75], and GPT-4 [76] are examples of how increasing the number of parameters can lead to significant gains in performance.

In this thesis, we leverage the capabilities of LLMs. Consequently, we delve into the technical specifics of the Transformer architecture, a fundamental component of LLMs, in the following section 2.1.1.

2.1.1 Transformer Architecture

The core component of the LLMs is the Transformer architecture [2], as shown in Figure 2.1. A typical transformer adopts encoder-decoder architecture, where the encoder and decoder both contain N identical blocks stacked together. In detail, given an input sequence $(x_1, x_2, ..., x_n)$, the encoder's role is to interpret and contextualise it. This is realised by converting the input into a set of contextualised representations or embeddings, denoted as $\mathbf{h} = (h_1, h_2, ..., h_n)$. Upon receiving these embeddings, the decoder then incrementally produces the output sequence $(y_1, y_2, ..., y_m)$. Particularly, the generation of each token in this sequence is informed both by the embeddings $\mathbf{h}$ and the previous generated token.

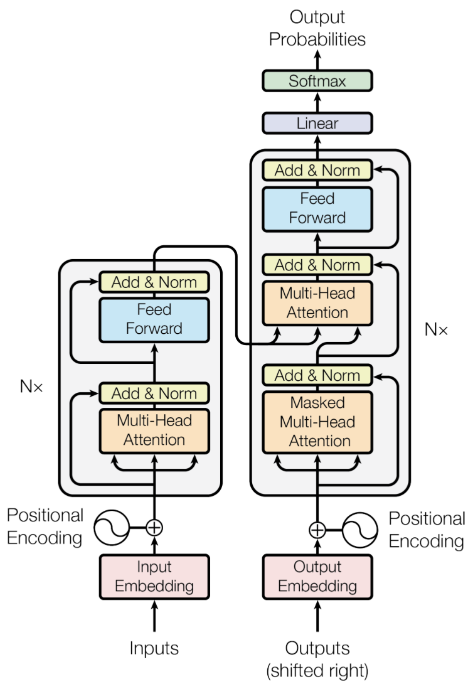


Figure 2.1: The Transformer architecture, image taken from the [2].

2.1.1.1 Encoder

As depicted in Figure 2.1, the encoder of the transformer architecture consists of N identical blocks, where each block is made up of a multi-head attention mechanism and a feed-forward neural network (FFN) layer, both equipped with residual connections [6] and followed by layer normalisation [77]. Specifically, for an input $\mathbf{x}_{\text{enc}} \in \mathbb{R}^{n \times h}$, it first undergoes the multi-head attention process, then after a residual connection and layer normalisation becomes $\mathbf{o}_{\text{multi\_head}}$. This output then passes through the FFN layer, following another round of residual connection and layer normalisation produces the final output $\mathbf{o}_{\text{enc\_block}}$, which serves as the input for the subsequent block. After traversing all blocks, the final encoder output, a contextualised representation captures complex patterns and long-range dependencies of the input:

$$\begin{aligned}\mathbf{o}_{\text{multi\_head}} &= \text{LayerNorm}(\mathbf{x}_{\text{enc}} + \text{MultiHead}(\mathbf{x}_{\text{enc}})) \\ \mathbf{o}_{\text{enc\_block}} &= \text{LayerNorm}(\mathbf{o}_{\text{multi\_head}} + \text{FFN}(\mathbf{o}_{\text{multi\_head}}))\end{aligned} \tag{2.1}$$

where MultiHead and FFN refer to the multi-head attention layer and feed-forward neural network layer inside one transformer block.

2.1.1.2 Decoder

Unlike the encoder, the decoder of the transformer architecture begins with a **masked** multi-head attention layer. This masking ensures that the prediction at position i is only influenced by positions preceding it. Following this layer, there's another multi-head attention mechanism that operates over the encoder's output, enabling the decoder to attend to the relevant parts of the input sequence. The decoder then employs a feed-forward neural network (FFN) supplemented with a residual connection and followed by layer normalisation. In essence, for a given encoder output $\mathbf{h} \in \mathbb{R}^{n\times h}$ and a decoder input $\mathbf{x}_{\text{dec}} \in \mathbb{R}^{l\times h}$ where l denotes the length of the decoder's output tokens thus far, the process can be summarised as passing through the masked attention, then the attention over the encoder's output, and finally through the FFN, generating a sequence of output tokens:

$$\begin{aligned}
&\mathbf{o}_{\text{masked\_multi\_head}} = \text{LayerNorm}(\mathbf{x}_{\text{dec}} + \text{MaskedMultiHead}(\mathbf{x}_{\text{dec}})) \\
&\mathbf{o}_{\text{multi\_head}} = \text{LayerNorm}(\mathbf{o}_{\text{masked\_multi\_head}} + \text{MultiHead}(\mathbf{h}, \mathbf{o}_{\text{masked\_multi\_head}})) \\
&\mathbf{o}_{\text{dec\_block}} = \text{LayerNorm}(\mathbf{o}_{\text{multi\_head}} + \text{FFN}(\mathbf{o}_{\text{multi\_head}}))
\end{aligned} \tag{2.2}$$

where MaskedMultiHead refers to the masked multi-head attention layer.

2.1.1.3 Multi-Head Attention

The strength of the transformer lies in its ability to leverage attention mechanisms, which allow it to weigh the importance of different input tokens differently. The attention function operates by generating three types of vectors: query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$). Initially, the attention function calculates the attention weights by computing the dot product between $\mathbf{Q}$ and $\mathbf{K}$. The attention weights are then applied to compute a weighted sum over the $\mathbf{V}$.

The scaled dot-product attention mechanism, as depicted in Figure 2.2, begins by

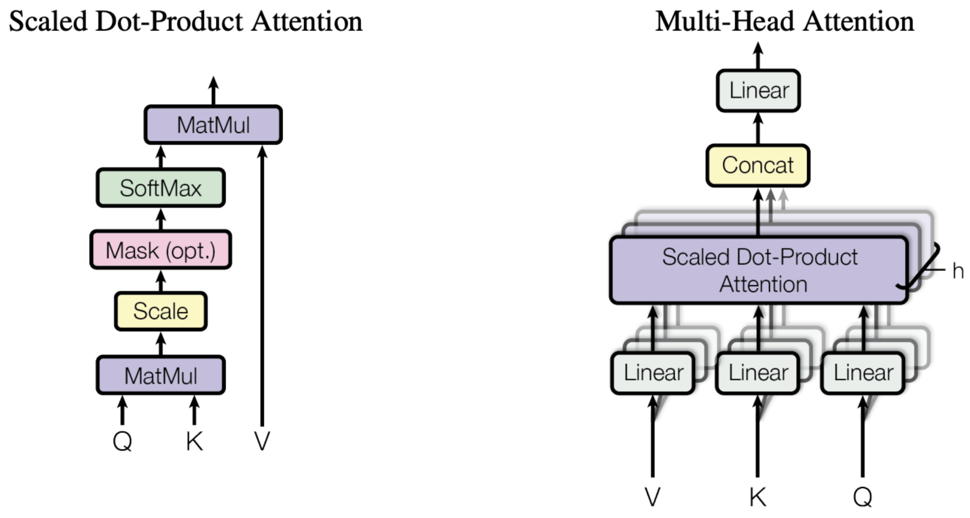


Figure 2.2: The left is the Scaled Dot-Product calculation, and the right is the Multi-Head version. The image is taken from [2].

taking the dot product of a query vector $\mathbf{q}$ with the entire set of key vectors $\mathbf{K}$. This output is subsequently normalised using the softmax function. Moreover, to mitigate the risk of exceedingly small gradients, the dot product between $\mathbf{q}$ and $\mathbf{K}$ is scaled down by dividing it by the square root of the dimensionality of $\mathbf{K}$:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{2.3}$$

where d_k is the dimension, $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, $\mathbf{K} \in \mathbb{R}^{m \times d_k}$, $\mathbf{V} \in \mathbb{R}^{m \times d_v}$ are the matrix version of query, key, and value vectors.

Expanding upon the scaled dot-product attention, the concept of multi-head attention is introduced. This approach employs multiple heads, each of which independently projects the query, key, and value vectors into distinct sub-spaces. As a result, each head is able to extract information from various aspects of the input data. Beyond the ability to obtain rich representations, multi-head attention also enhances computational efficiency when running on GPUs. The output from the multi-head attention mechanism is computed as:

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{CONCAT}(\text{softmax}(\frac{\mathbf{Q}\mathbf{W}_i^Q(\mathbf{K}\mathbf{W}_i^K)^T}{\sqrt{d_k}})\mathbf{V}\mathbf{W}_i^V)\mathbf{W}^o \tag{2.4}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model}\times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}}\times d_k}$, and $\mathbf{W}_i^V \in \mathbb{R}^{d_{model}\times d_v}$ are projection weights for the i-th head. h here is the number of heads, and $\mathbf{W}^o \in \mathbb{R}^{hd_v\times d_{\text{model}}}$ project the dimension of concatenated multi-head output back to the target dimension d_{model}.

2.1.2 Types of Large Language Models

LLMs can be broadly classified into three categories: encoder-only, decoder-only, and encoder-decoder models.[1]

- **Encoder-only LLMs**: These utilise only the encoder component of the transformer architecture (detailed in Section 2.1.1.1). They are often termed auto-encoding models because they primarily serve to embed input data into dense representations. With the use of self-attention modules (as discussed in Section 2.1.1.3), these models can bi-directionally access the complete input sequence at every step. Well-known encoder-only LLMs include BERT [78], RoBERTa [79], ALBERT [80], and DistilBERT [81], etc.

- **Decoder-only LLMs**: LLMs in this category function as decoders for generative tasks. Examples of such models are CTRL [82], GPT-2 [83], Transformer XL [84], and LLaMA [75], and GPT-3 [74]. These models incorporate the decoder segment of the Transformer architecture (explained in Section 2.1.1.2). In contrast to encoder-only LLMs, when a word is inputted into the self-attention modules in these models, it can only access the tokens that come before it. This method of generation is referred to as auto-regressive generation.

- **Encoder-decoder LLMs**: These models employ both the encoder and decoder portions of the Transformer. While the encoder can access every token in a

[1]To clarify and define terms succinctly, we integrate pre-trained language models with large language models in this thesis.

sequence at each stage, the decoder can only access preceding tokens at that stage. Prominent encoder-decoder LLMs are BART [85], mBART [86], and T5 [87].

2.1.3 Pre-training Large Language Models

Currently, there is an vast amount of data available online, but annotating this data for specific tasks to train deep-learning models is both labour-intensive and time-consuming. However, LLMs offer a solution to this challenge by allowing training on this unlabelled data through unsupervised learning, a process commonly referred to as **pre-training**. Pre-training enables LLMs to derive meaningful representations from large datasets, which can then be utilised for specific downstream tasks.

The method of pre-training is crucial for LLMs to acquire effective representations. One prevalent approach involves corrupting the input, such as by randomly masking certain words. The LLMs are then tasked with reconstructing or predicting these masked words, as seen in models like BERT [78] and ALBERT [80]. Another method is the auto-regressive task, where the objective is to forecast subsequent words based on preceding ones in a sentence. This auto-regressive pre-training is particularly apt for decoder-only LLMs, as exemplified by models like GPT-2 [83] and GPT-Neo [88]. Some more advanced pre-training strategies include objectives like predicting an entire masked segment of text rather than just individual words, as implemented in models such as T5 [87].

2.1.4 Task-specific Fine-tune Large Language Models

The sheer size of LLMs results in a vast number of parameters, making the pre-training process both computationally expensive and data-intensive. Given these challenges, it is often not feasible for many researchers and organisations to train an LLM from the ground up. A more practical approach that has gained traction recently is to utilise pre-trained LLMs as a starting point, initialising them with weights derived from prior training, and then further refining these models on task-specific datasets. This process is commonly referred to as "fine-tuning".

Fine-tuning methods can be broadly categorised into two primary approaches. The first approach involves keeping the pre-trained LLM's parameters fixed, while adding a supplementary, lightweight layer on top. During the fine-tuning phase, only the parameters of this additional layer are updated, leaving the original LLM weights unchanged [89]. The second method entails updating the weights of the entire LLM, including any added layers. However, a crucial point to note in this approach is that a reduced learning rate, compared to the original pre-training phase, is typically used. This ensures that the LLM retains the knowledge it gained during pre-training and avoids drastic shifts in the parameter space [84, 90, 91].

2.1.5 In-Context Learning

As the size of LLMs continue to expand, it has become evident that larger models exhibit enhanced capabilities [92, 93]. One of the most significant emergent abilities in these scaled-up LLMs is "in-context learning". This capability enables the model to absorb and respond to natural language instructions, often referred to as "prompts", without any specialised fine-tuning for a given task. By providing the model with one or just a handful of these prompts, it can produce the expected output or answer, aligning with the instruction given. This approach, which bypasses the need for extensive task-specific fine-tuning, is termed one-shot or few-shot learning.

The importance of in-context learning in the research community cannot be understated. Given the enormous size of these models, the computational cost of fine-tuning them has become prohibitive for many. Thus, the ability to leverage them effectively with only a few prompts presents a significant advantage. This prower of in-context learning is evident in some of the latest giant LLMs such as PaLM [94], Bloom [95], Galatica [96], Bard [97], and GPT-4 [76].

For smaller LLMs, complex reasoning spanning multiple steps, especially in mathematical domains, can be challenging. The chain-of-thought (CoT) technology [98] provides a solution, enhancing the LLMs' capability to handle such tasks. With CoT, prompts contains demonstrations of solving the problems by breaking down the reasoning processes into manageable steps can guides LLMs to mimic this solving process, which finally improve reasoning accuracy. This approach offers better problem de-composition, allows for self-correction [99], and enhances interpretability by producing intermediate steps. Furthermore, it makes training more efficient and offers potential extensibility to other domains. CoT has shown marked improvements, especially in areas like code generation [100] and solving maths word problems [101–104].

In this thesis, encoder-only LLMs are utilised as robust encoders to distil the intricate semantic essence from mathematical problem statements in Chapter II. Concurrently, given that the gap between the textual narrative and geometric diagrams is bridged via the conversion of diagrams into natural language in Chapter 6, decoder-only LLMs present themselves as ready-to-use decoders, facilitating the generation of solution programs.

2.2 Text-based Automated Numerical Reasoning Overview

This section delves into a thorough review of literature over text-based numerical reasoning. Initially, we outline established benchmarks for the evaluation of mathematical reasoning capabilities in Section 2.2.1. This includes datasets designed for maths word problems (MWPs) and those for maths question answering (MATH-QA) tasks. Following this, Section 2.2.2 offers a detailed examination of various methods developed within this domain, ranging from rule-based approaches to statistical techniques, and evolving through to Sequence-to-Sequence/Tree-based methodologies, concluding in the exploration of methods based on large language models (LLMs).

2.2.1 Benchmarks for Text-based Numerical Reasoning Task

2.2.1.1 Datasets of Maths Word Problems

As we discussed in Section 1.2.1, maths word problems (MWPs) challenge AI systems to generate equations that encapsulate the reasoning required to arrive at the correct answer, making them an excellent measure of the system's mathematical reasoning skills. Several foundational datasets have been introduced, such as A12/Verb395 [32], ALG514 [33], IL [34], and SinEQ [30], etc. A12/Verb395 is notably the first dataset designed for training models to tackle MWPs, consisting of 400 problems that are annotated with arithmetic equations focused on addition and subtraction operations. These early datasets were typically collected from crowd-sourced platforms; for instance, ALG514 was sourced from "Algebra.com".

The development of deep-learning methods has indeed revolutionised the approach to solving mathematical word problems (MWPs). However, the full potential of these methods can only be explored with access to sufficiently large and high-quality datasets. The creation of Math23K [105], a pioneering dataset in the field, marked a significant advancement in this direction. Comprising 23,161 problems with annotated equations represented in logical forms. Later, datasets like HWMP [106] have been introduced. HWMP caters to more complex problem-solving by including problems with several unknown variables, thus pushing the boundaries of what deep-learning models can achieve. Others using natural languages to annotate the reasoning process, Wang et al. [107] proposed the use of natural language rationales to describe the reasoning behind solutions. This approach is popular and used in the construction of datasets such as GSM8K [36], which consists of high-quality MWPs at a grade school level. However, while natural language offers flexibility, it can sometimes lack the precision that mathematical reasoning demands. To mediate between the formality of equations and the flexibility of natural language, Amini et al. [9] introduced the MathQA dataset. The introduction of such datasets is crucial in challenging and refining the capabilities of AI in mathematical reasoning, aiming to closely simulate the nature of mathematical reasoning ability of human beings.

2.2.1.2 Datasets for Maths Question Answering

The task of maths question answering (MATH-QA) extends beyond maths word problems (MWPs) task. It requires the AI system to contain the in-depth understanding of mathematical reasoning ability as well as an awareness of general and subject-specific knowledge.

The DROP [59] and Mathematics [29] datasets are notable in the maths question answering task, distinct from typical MWPs datasets that focus on equations as reasoning process. They feature a variety of answer formats, including text spans, numerical counts, etc.

Just as with MWPs, using datasets focused on specific domains might lead to an overestimation of the mathematical reasoning capabilities of AI systems. To provide a more comprehensive assessment, the research community has begun creating datasets that amalgamate problems from various domains, such as the Lila dataset [108], which incorporates 23 distinct mathematical reasoning tasks.

The task of MATH-QA extends beyond purely textual data to include tables, a format prevalent in various real-world contexts like scientific research, medical records, financial statements, etc. As such, proficiency in resolving mathematical problems within these hybrid data forms is essential. To advance research in this area, datasets such as FinQA [10] and TAT-QA [109] have been constructed from the financial reports.

In the field of MATH-QA, the large variety of problem types typically results in datasets that only offer the final answer without any annotations of the intermediate reasoning processes. This presents a significant challenge for effectively leveraging such datasets for model training and evaluation. To counter this obstacle, recent research has been making efforts into depicting the reasoning process through various forms such as logical expressions [10, 26, 28], solution programs [110], or descriptive natural language explanations [108], thereby enhancing the datasets' utility for developing AI systems with more robust mathematical reasoning ability.

Table 2.1 provides a comprehensive list of the datasets created for both the MWPs and MATH-QA tasks. It details the format of annotations, the size of the data, and the year that each dataset was proposed.

2.2.2 Methods for Text-based Numerical Reasoning Task

In this section, we describe the evolution of methods aimed at text-based numerical reasoning. Initially, we briefly introduce rule-based methods predominantly developed in the previous century. Following that, we explore the statistical approaches that have been applied to this challenge. Finally, we give a thorough examination of deep-learning based methods, with particular attention to Sequence-to-Sequence/Tree architectures and methods leveraging Large Language Models (LLMs).

Name	Task	Annotation	Size	Year
A12/Verb395 [32]	MWPs	Arithmetic Equation	400	2014
ALG514 [33]	MWPs	Arithmetic Equation	514	2014
IL [34]	MWPs	Arithmetic Equation	1,404	2015
SingleEQ [30]	MWPs	Arithmetic Equation	508	2015
DRAW-1K [111]	MWPs	Arithmetic Equation	1,000	2015
MAWPS [112]	MWPs	Arithmetic Equation	3,320	2016
Dolphin18K [113]	MWPs	Arithmetic Equation	18,460	2016
AllArith [114]	MWPs	Arithmetic Equation	831	2017
Math23K [105]	MWPs	Arithmetic Equation	23,162	2017
Aqua [107]	MWPs	Natural Language	100,000	2017
Aggregate [115]	MWPs	Arithmetic Equation	1,492	2018
MathQA [9]	MWPs	Solution Program	37,292	2019
QUAREL [26]	MATH-QA	Logic Form	2,771	2019
McTaco [27]	MATH-QA	Direct Answer	13.225	2019
DROP [59]	MATH-QA	Direct Answer	96,597	2019
Mathematics [29]	MATH-QA	Direct Answer	2,010,000	2019
HWMP [106]	MWPs	Arithmetic Equation	5,470	2020
ASDiv [116]	MWPs	Arithmetic Equation	2,305	2020
Fermi [28]	MATH-QA	Solution Program	11,000	2020
SVAMP [31]	MWPS	Arithmetic Equation	1,000	2021
GSM8K [36]	MWPS	Natural Language	8,792	2021
MathQA-Python [117]	MWPs	Python Program	23,194	2021
FinQA [10]	MATH-QA	Solution Program	8,281	2021
Math [18]	MATH-QA	Natural Language	125,000	2021
TAT-QA [109]	MATH-QA	Direct Answer	16,552	2021
MultiHertt [110]	MATH-QA	Arithmetic Equation	10,440	2022
NumGLUE [21]	MATH-QA	Direct Answer	101,835	2022
Lila [108]	MATH-QA	Python Program	134,000	2022

Table 2.1: Table listing all introduced datasets for MWPs and MATH-QA tasks.

2.2.2.1 Rule-based and Statistical Methods

The research on endowing AI with the capability of mathematical reasoning has been attracted the interest for decades [24, 25, 118, 119]. In the early stages, researchers primarily utilised rule-based methods to transform text descriptions into structured representations through techniques like pattern matching, verb categorisation, or semantic parsing [21, 23, 120, 121].

However, the early rule-based methods, which involved associating maths problems with structured representations, heavily relying on manually crafted patterns which limited their ability to generalise to new problems. Hence, researchers shifted focus towards statistical models capable of autonomously identifying patterns in training data. One of the key advancements in this direction was by Kushman et al. [33], who introduced a method that models the joint probability distribution of MWPs and their corresponding target equations. Subsequent research efforts concentrated on enhancing the accuracy of template extraction. For instance, Hosseini et al. [32] hypothesised that verbs are crucial for identifying relevant equations and thus focused on learning word representations by categorising verbs in the problem statements. There were also studies that emphasised the role of nouns and numbers in the text [122, 123], all aiming to improve the precision of mapping from textual descriptions to mathematical expressions.

The approaches mentioned above typically involve representing the reasoning process through mathematical equations. However, some researchers like Koncel-Kedziorski et al. [30] and Roy et al. [123] opted for a different representation strategy by transforming these mathematical equations into expression trees. The principal benefit of using expression trees is that they inherently prevent the generation of invalid mathematical equations.

2.2.2.2 Deep Learning Methods: Sequence-to-Sequence/Tree Methods

Sequence-to-Sequence The Sequence-to-Sequence (Seq2Seq) architecture [124], has gained significant traction across a range of generation tasks, including machine translation [125, 126], text summarisation [127, 128], image captioning [129, 130], and story generation [131, 132]. These diverse tasks are unified by a common feature: the necessity to craft outputs that reflect a deep comprehension of contextual information. All tasks share the same feature that they need to generate outputs according to the understanding of the context. Consequently, the framework has also been adopted for resolving mathematical reasoning problems. In this context, the standard Seq2Seq model designated for mathematical reasoning typically adopts the encoder-decoder schema. Within this structure, the encoder is tasked with processing the input maths problem text, while the decoder is responsible for producing the corresponding output, often in the form of an arithmetic equation or a similar program structure. The selection of neural network architectures for both the encoder and decoder components is varied, with prevalent choices including Long Short-Term Memory (LSTM) networks [133], Gated Recurrent Units (GRU) [134], Graph network [135], and transformer architecture [2]. Each of these choices brings a unique set of strengths to the model's mathematical reasoning ability.

DNS model [105] is a pioneering Seq2Seq approach, specifically for addressing Math

Word Problems (MWPs). DNS introduces a novel strategy aimed at curtailing the breadth of the search space for potential mathematical equations by implementing a normalisation technique for the numbers present in both the problem texts and the target equations. This technique encompasses two key steps: firstly, it substitutes numerical values in the text with their positional indices, and secondly, it replaces the numerical values in the target equations with a sequence reflecting the order of their indices. By fixing the size of the decoding vocabulary, this normalisation method has been integrated into subsequent work [136–140]. Building on this, Ling et al. [141] presented an innovative staged back-propagation method. Unlike its predecessors that focused on equation generation, they adopt rationals as the output, thereby improving the model's interpretability. Further evolving the field, Robaidek et al. [142] shifted the complexity of the generation task towards a classification-oriented task. Their approach involved the development of multiple classifiers, each trained to select the most fitting output from a range of outputs produced by different models.

The prevalent normalisation method of substituting numbers with their positional indices has facilitated the reduction of the search space in decoding out the target equations. Nevertheless, this method has its drawbacks, notably the loss of intrinsic numerical semantics, which can lead the models to make erroneous operational decisions. An illustrative example of such an error would be an inappropriate attempt to aggregate "the price of two apples" with the "number of apples," which conceptually mismatches the distinct semantic roles that numbers play in problems. Addressing this critical issue, Chiang et al. [138] have introduced an encoder-decoder framework that retains the numerical semantics by capturing the contextual meanings of numbers from the problem text. Their approach is characterised by an encoder that is specifically designed to discern the semantic implications behind each number within the text. Complementing this, the decoder incorporates a stack mechanism which serves to maintain and track the semantic significance of the operands throughout the decoding process.

Previous approaches within the Seq2Seq framework conventionally generate target maths equations in a sequential manner. This method, however, inadvertently leads to a non-deterministic output space owing to the production of multiple, equivalent target programs. For instance, the equations "$1+2+3$" and "$1+3+2$" are mathematically equivalent, yet they are generated as distinct outputs by the model. To address this redundancy, subsequent research has proposed representing equations as expression trees [136,137] . In this representation, both operators and numerical values are treated as nodes within a tree structure. This naturally embodies the hierarchical nature of maths equations. The models are then trained to generate the post-order traversal of these expression trees.

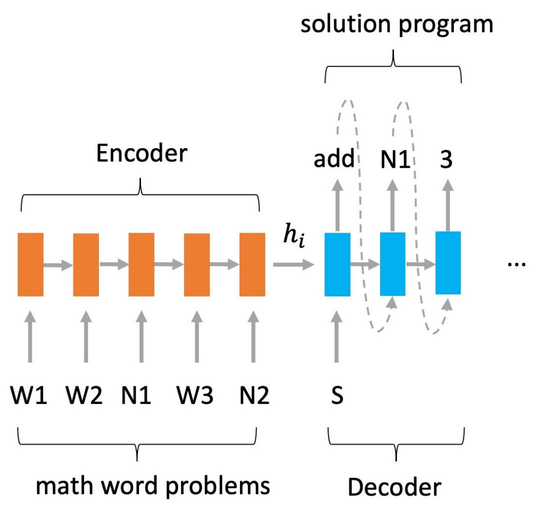


Figure 2.3: The Seq2Seq model for mathematical reasoning task. Here, **W** represents the textual words in maths problem texts and **N** signifies the numerical elements within the problem texts. The encoder converts each word in the text into a hidden state vector **h**. These vectors are subsequently utilised as the initial input alongside a start token **S** for the decoder. The decoder then generates an output at each step, which serves as the subsequent input for the following step in the decoding process.

Figure 2.3 depicts a fundamental Seq2Seq architecture tailored for mathematical reasoning tasks. The model processes an input maths problem text, denoted as **W1, W2, N1, ...**, where **W** stands for words and **N** for numbers. The encoder transforms these inputs into vector representations **h**, capturing the understanding of the input sequence. Following this, the decoder employs the vectors **h** as its initial input to sequentially produce the solution program.

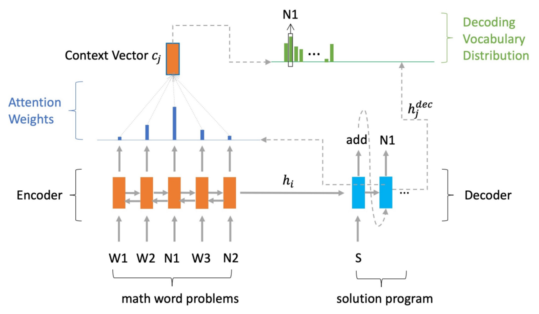


Figure 2.4: The Seq2Seq model incorporating attention for mathematical reasoning task. During the decoding phase for step j, the decoder's hidden state h_j^{dec} engages in the computation of attention scores in relation to all the encoder's hidden states. These attention scores are then applied to perform a weighted summation of the encoder's hidden states, resulting in the context vector c_j. This context vector, along with the hidden state h_j^{dec}, contributes to the generation of the probability distribution across the decoding vocabulary. From this distribution, the word "N1" is selected for having the highest probability.

Sequence-to-Sequence with Attention Mechanism The initial Seq2Seq models proposed for mathematical reasoning tasks encountered substantial limitations due to their reliance on encoding all the information from the problem text into a fixed-size vector. This constraint posed a significant challenge, as it required the model to compress all the necessary details into a single representation, regardless of the problem's complexity or length, which could lead to a loss of information and context critical for solving mathematical problems. To enhance the capability of the encoder-decoder architecture in Seq2Seq models, the attention mechanism was introduced [143]. This innovation allows the decoder to attend to different segments of the input text during each step of the output generation process. At each step of the output generation, the model can weigh the importance of different parts of the input, focusing on the most relevant information at that step.

Utilising the attention mechanism, we are able to identify a broader specialised features for tackling mathematical problems. Li et al. [139] have introduced four addition features: (1) global features that provide a holistic view of the problem text; (2) quantity-related features that delineate the association between numerical values and their contextual terms; (3) quantity-pair features that model the inter-numeric relationships; and (4) question-related features that define the connections between numerical data and the question posed. These comprehensive features effectively minimise the influence of noisy data, refining the model's performance.

Figure 2.4 illustrates a standard Seq2Seq model with an attention mechanism, tailored for tasks in mathematical reasoning. This model is different from the conventional Seq2Seq model through the computation of the context vector c_j at each j-th step of output generated by the decoder. First, attention weights $attn_j$ are calculated, reflecting the relevance of each word in the maths problem text to the decoder's hidden state h_j^{dec} at the specific time step j:

$$\begin{aligned} e_i^j &= W\tanh(W_h h_i + W_{dec} h_j^{dec} + b_{attn}) \\ attn_j &= \text{softmax}(e^j) \end{aligned} \tag{2.5}$$

where $W \in \mathbb{R}^{1\times h}$, $W_h \in \mathbb{R}^{h\times h}$, and $W_{dec} \in \mathbb{R}^{h\times h}$ are trainable parameters. The b_{attn} is the bias. Subsequently, the context vector c_j is computed by aggregating all of the encoder's hidden states, each weighted by their corresponding attention weight:

$$c_j = \sum_i attn_j^i h_i \tag{2.6}$$

Lastly, the probability distribution over the decoding vocabulary, denoted as $\mathbb{P}_{vocab}$, is determined by concatenating the context vector c_j with the current hidden state from the decoder:

$$\mathbb{P}_{vocab} = \text{softmax}(W_{vocab}[h_j^{dec}, c_j] + b_{vocab}) \tag{2.7}$$

where $W_{vocab} \in \mathbb{R}^{|V|\times h}$ and b_{vocab} are trainable parameters and bias, respectively. $|V|$ is the size of the decoding vocabulary.

The application of attention mechanisms has made a significant impact in mathematical reasoning, particularly with the advent of pointer network architecture [144, 145]. Pointer networks are adept at overcoming the constraints of conventional Seq2Seq models where the output vocabulary is predefined and limited. These networks excel in tasks where the output elements must be dependant on the input sequence — a common scenario in mathematical reasoning, where numbers or variables from the problem text often form components of the solution.

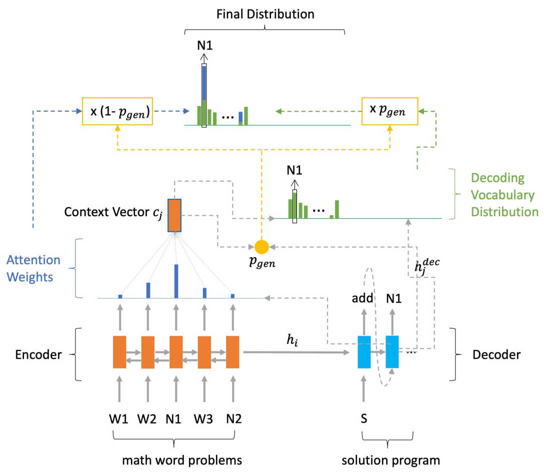


Figure 2.5: The Seq2Seq with Pointer-Network for mathematical reasoning task. During the decoding phase at time step j, the model computes the weight p_{gen} for the generation probability using the decoder's hidden state h_j^{dec} in conjunction with the context vector c_j. Subsequently, the decoding vocabulary's probability distribution is adjusted by adding the product of attention weights multiplied $(1 - p_{gen})$. The word "N1", which holds the maximum probability within the final distribution, is then chosen as this time step's output.

Figure 2.5 presents the integrated architecture of a Seq2Seq model with a Pointer-Network. It illustrates that, at time step at time step j, the predicted probability distribution for the decoding vocabulary $\mathbb{P}_{vocab}^{final}$ is derived from a combination of attention weights $attn_j$ and the base vocabulary probabilities $\mathbb{P}_{vocab}$:

$$\begin{aligned} p_{gen} &= \sigma(W_1 c_j W_2 h_j^{dec} + b_{gen}) \\ \mathbb{P}_{vocab}^{final} &= p_{gen}\mathbb{P}_{vocab} + (1 - p_{gen})attn_j \end{aligned} \tag{2.8}$$

where $W_1 \in \mathbb{R}^{1\times h}$ and $W_2 \in \mathbb{R}^{1\times h}$ are trainable parameters. b_{gen} is the bias. σ refers to the sigmoid function.

Sequence-to-Tree While transforming equations into post-order traversal sequences of their expression trees is a strategy to resolve invalid equations generated by Seq2Seq models, this process does not entirely eliminate the production of invalid formulas [146].

To address this issue, some researchers have attempted to encode additional auxiliary information, such as the sibling node at each step, to learn the implicit tree structure within the sequence. However, these solutions have proven to be overly intricate. In response to the complexity of these methods, Liu et al. [147] proposed a approach by transforming the equation into an Abstract Syntax Tree (AST). This AST effectively encapsulates the hierarchical structure of mathematical equations, providing a clear and structured representation that can mitigate the generation of invalid equations. For a visual illustration, Figure 2.6 exemplifies the distinct differences in representing an equation through various formats.

Problem:	A slow car drives $58(n_1)$ km/h, and a fast car drives $85(n_2)$ km/h. The two cars drive at the same time in inverse direction, and they meet after $5(n_3)$ hours. How many kilometers does the fast car drive more than the slow car when they meet?	**AST**:
Equation:	$(n_2 - n_1) \times n_3$	
Prefix:	$\times - n_2 n_1 n_3$	
Suffix:	$n_2 n_1 - n_3 \times$	
Answer:	135	

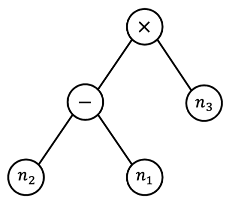


Figure 2.6: Different representations of the equation for a MWP. Prefix/Suffix refers to the pre/post order traverse of the AST. The figure is from [3].

To generate the AST as the target to resolve mathematical reasoning task, researchers have proposed the Seq2Tree model [146–148], which is adept at capturing the hierarchical tree-structured relationships inherent in Abstract Syntax Trees (ASTs). This approach extends the capabilities of traditional Seq2Seq models by enabling the generation of outputs that are structured as trees rather than linear sequences, aligning more closely with the intrinsic structure of mathematical equations.

Seq2Tree approaches, exemplified by the GTS model [146], have demonstrated promising results in the context of mathematical reasoning tasks. Nonetheless, Zhang et al. [149] identified limitations in the generalisability of models that rely on Abstract Syntax Tree (AST) representations. For instance, commutative operations like "3+2" and "2+3" produce the same result and should both be considered correct, but a Seq2Tree model may erroneously mark one as incorrect due to the strict hierarchical structure of AST. To rectify this issue, the concept of a teacher-student network was introduced [149]. This framework allows for a more flexible interpretation of mathematical expressions by enabling the student model to learn multiple valid representations of an equation from the teacher model.

Limitations of Sequence-to-Tree Models Despite the progress achieved by mentioned methods in addressing mathematical reasoning tasks, these models do indeed have several limitations:

1. **Train Test Discrepancy**: In the mathematical reasoning task, Seq2Seq/Tree models are conventionally trained using MLE, optimising the prediction of each subsequent token based on the preceding ones. This approach, while aligning with gradient descent due to its differentiable nature, introduces a misalignment with the testing phase which evaluate performance based on accuracy — a non-differentiable metric. This training-testing discrepancy leads to models that excel in training environments but may stumble when encountering the complex and unpredictable scenarios of real-world application [150].

2. **Lacking External Knowledge**: The incorporation of external knowledge into Seq2Seq/Tree models for mathematical reasoning has been limited, often constrained to the addition of constant tokens to the decoding vocabulary. This approach only allows for a restricted set of knowledge to be used, which can be a significant limitation when solving complex problems that require broader knowledge. Addressing this gap, Wu et al. [151] introduced the Knowledge-Aware Sequence-to-Tree (KA-S2T) network. This innovative model extends beyond the confines of predefined constants by linking entities directly with an external knowledge base. By doing so, it dynamically integrates a richer set of knowledge into the problem-solving process, enhancing the model's ability to tackle a wider variety of mathematical reasoning tasks.

3. **Not Using the Real Number Value**: Prior methods in mathematical reasoning often employ a normalisation strategy where numbers within a problem are replaced with index, and pointer networks are utilised to select these indices for

the output, discarding the actual numerical values. This technique ignores the semantic content numbers carry — their magnitude, relationships, and the role they play in mathematical operations. Without considering the true values, models may fail to grasp the deeper meaning behind the quantities involved, potentially leading to solutions that are structurally correct but semantically flawed [152].

Addressing the above limitations, there has been significant progress through various innovative approaches. Hong et al. [153] presented the Learning-By-Fixing (LBF) framework, which narrows the gap between training and testing objectives using policy gradient methods. LBF adapts to incorrect predictions by employing a "fixing" mechanism, similar to human learning through error correction. Qin et al. [154] introduced auxiliary tasks with the Number Solver (NS-Solver) to harness the full potential of the training data. These tasks encompass predicting numbers, identifying commonsense constants, checking program consistency, and exploiting mathematical duality, each contributing to a more robust learning process.

Additionally, traditional Seq2Seq and Seq2Tree methods, with their left-to-right and top-down decoding strategies, respectively, are challenged when required to generate multiple target programs due to their inherent sequential processing. Cao et al. [155] tackled this by proposing a bottom-up approach with the Seq2DAG model, allowing for simultaneous generation of multiple target programs.

Graph-to-Tree The approaches proposed for mathematical reasoning tasks have seen merely developing sophisticated decoders to generating target outputs, without considering to enhance the encoding of problem texts to better capture their intrinsic information. Prior Seq2Seq/Tree methods predominantly concentrated on the generation of target programs, often encoding the problem simply as a sequence of words represented by a singular vector. This approach falls short in emulating the human-like comprehension of problems, given the complexity and richness of information contained [156, 157].

To bridge this gap, researchers have begun incorporating graph-based modules to represent the problem text more effectively. These methods aim to encapsulate not only the sequence of words but also the multifaceted relationships and structures within the problems. Lin et al. [156] introduced the HMS, which utilises a graph module to deeply analyse the relationships between the entire problem text and its entities, providing a more profound understanding and utilisation of information to solve MWPs. Similarly, innovations like the Multi-Encoder/Decoder [3] and Graph2Tree [158] further expand on this approach, leveraging the power of graphs to model the detailed interconnections within problem texts, thereby enabling models to approach human-level problem-solving capabilities.

2.2.2.3 Large-Language-Models-based Methods

The utilisation of large language models (LLMs) has resulted in notable successes across a diverse range of applications [78–80, 83, 85, 87]. Nevertheless, when it comes to mathematical reasoning, these models tend to fail. The underlying factors for this shortfall are diverse. Initially, the pre-training of these models typically does not involve many mathematical texts or problems. Additionally, the pre-training objectives used in LLMs, such as masked and causal language modelling, are not specifically designed to learn the skills necessary for mathematical reasoning. Finally, the effectiveness of LLMs depends on the availability of substantial datasets for training, yet there is a conspicuous scarcity of such expansive mathematical datasets. Consequently, there has been a shift toward refining LLMs for mathematical reasoning through targeted fine-tuning on specific tasks, as such method requires less dataset. Kim et al. [159] pioneered this approach by integrating a pointer network with the transformer model architecture to enhance problem-solving capabilities in mathematics. Furthermore, other researchers have leveraged the extensive linguistic capabilities inherent in LLMs to better interpret the textual content of maths problems [160]. The inherent flexibility of LLMs has also inspired researchers to extend their use beyond text-based mathematical reasoning tasks. Emerging studies have begun tackling mathematical problems presented in the tabular format [10, 109, 110] and the image format [161].

Directly fine-tuning LLMs for mathematical reasoning tasks is hindered by challenges comparable to those faced by Seq2Seq/Tree approaches. A notable challenge is the performance drop observed as the length of the solution program increases [162]. To resolve this issue, Shen et al. [163] introduced a "generate and rank" approach, incorporating an additional ranking mechanism specifically to discern correct from incorrect solutions. This concept of evaluating and ordering the outputs generated by various models has been further developed in subsequent research [36, 162].

Moreover, when fine-tuning LLMs for tasks involving mathematical reasoning, prevailing techniques typically depend on the MLE objective, which focuses on maximising the probability of the target solution program. Yet, this approach does not account for the fact that many mathematical problems can have several correct solutions, each representing a different logical path to the same output. Training models without this variety of correct solution paths can lead to overfitting, resulting in diminished model performance on new, unseen problems [164]. To address this, Ni et al. [164] have updated the fine-tuning process of LLMs by including a diversity of self-sampled solutions during training, equally treating these multiple correct solutions as potential target programs.

In addition to the fine-tuning of LLMs, alternative approaches have been explored to equip models with the capability for mathematical reasoning. Li et al. [165] argued that traditional methods tend to conflate the representations of maths problems that share the same solution, despite potentially significant semantic differences in their descriptions. To overcome this, they employed a contrast learning strategy that focuses on identifying and aligning similar problem prototypes. This method assists LLMs in recognising and distinguishing between various patterns, enhancing their ability to understand the underlying structures of maths problems.

Research has delved into why LLMs don't exhibit the same level of proficiency in mathematical reasoning as they do in other tasks. A common hypothesis is that the underlying Transformer architecture might be a limiting factor. The architecture typically processes reasoning tasks in a single forward pass, which can be inadequate for complex mathematical reasoning that requires multi-step calculations [166]. To address this, researchers tried to experiment with generating intermediate reasoning steps before arriving at the final answer [167]. This concept is the infancy of the future work that is called the “chain-of-thought” approach, a method that mimics human-like step-wise reasoning to tackle complex tasks. This chain-of-thought technique and its applications will be explored in the subsequent section.

LLMs often struggle with mastering mathematical reasoning ability from standard pre-training routines [28, 40]. For LLMs to effectively assimilate mathematical understanding that benefits downstream tasks, some researchers have introduced customised pre-training tasks aimed at injecting mathematical reasoning abilities directly into these models [168]. Peirce [169] identified three critical aspects essential for conducting reasoning: 1. Deduction: The process of inferring specific outcomes from general facts and established rules. 2. Induction: The capacity to formulate broad rules from an observed set of specific instances. Abduction: The skill of hypothesising a likely cause or rule given certain evidence or outcomes. Building on this, Wu et al. [170] crafted specialised synthetic tasks tailored to each of these reasoning types. They posited that LLMs could develop a strong inductive reasoning capability by training on these synthetic tasks, which in turn helps downstream tasks. Moreover, others in the field have tried to integrate certain mathematical reasoning ability into LLMs, such as magnitude and numeric type recognition [171]. This is proposed in the hopes that a better grasp of these fundamental numerical skills will benefit mathematically intensive reasoning tasks.

Recent research has suggested that auto-regressive pre-training of LLMs using a quantitative-relevant corpus can be sufficient for downstream mathematical reasoning tasks, demanding the need for vast amounts of training data [18, 49, 50, 53, 55, 172–174]. These specialised corpora, rich in quantitative content, are prevalent in fields closely associated with quantitative analysis, such as financial reports, scientific literature, and educational materials. For instance, BloombergGPT [175], an LLM with 50 billion parameters, was trained on a financial dataset comprising 363 billion tokens. This extensive pre-training enables it to perform a broad spectrum of tasks within the financial domain. Similarly, in the realm of science, Galactica [96] has been trained on an extensive scientific corpus that includes 48 million academic papers, textbooks, and lecture notes, as well as databases of millions of chemical compounds and proteins, scientific web content, and encyclopedias. This comprehensive pre-training allows the model to handle a wide array of data-rich, number-intensive tasks in scientific domains.

The initial wave of LLMs, such as RoBERTa [79] and BERT [78], contain around 123 million parameters. Successor models like T5 [87] and BART [85] marked a substantial increase in the model size, boasting up to 770 million parameters. Today, we witness a significantly expansion in the scale of LLMs, with models like GPT-3 [74] encompassing up to 175 billion parameters. As the parameter count in these LLMs has surged, it is hard to train such them due to the considerable computational resources required. However, one of the significant advantages of these larger models is their capacity for in-context learning, which allows LLMs to perform tasks by being fed examples directly at inference time [74, 176]. To utilise in-context learning for mathematical reasoning tasks, researchers began to provide LLMs with maths problems accompanied by their solution programs to guide the models. Nevertheless, LLMs often could not produce accurate answers consistent with the provided examples [94]. Insights from psychological research suggest that arriving at an answer directly is often not feasible. Instead, models may need to follow a sequence of intermediate steps [177]. Building on this understanding, Wei et al. [98] introduced the concept of a "chain-of-thought" (CoT), which presents LLMs with examples as inputs that include explicit step-by-step reasoning. The CoT approach has substantially enhanced the mathematical reasoning capabilities of LLMs, leading to marked improvements and notable achievements across a variety of mathematical reasoning benchmarks.

The process of manually crafting few-shot CoT examples, while useful, may not adequately address the full range of problem scenarios that an LLM might encounter. Recognising this limitation, some researchers have turned their focus toward devising methods for LLMs to autonomously generate high-quality CoT examples [178–181]. In one of these approaches, Lu et al. [19] introduced a technique that leverages policy gradient methods to refine the selection of in-context examples from a limited dataset for the in-context learning. This method allows the model to identify and utilise the most effective examples to guide its learning process. To further augment the mathematical reasoning capabilities of LLMs via CoT, other researchers have investigated ensemble strategies. These involve generating multiple solution programs and subsequently selecting the most accurate one. Ensemble methods exploit the diversity of problem-solving approaches inherent in LLMs to improve the chances of reaching the correct solution, effectively combining the strengths of various reasoning paths to enhance overall performance [101, 102].

As LLMs continue to expand in size, leveraging them even through in-context learning has become increasingly resource-intensive, with the cost of significant computational power, like GPUs, reaching into the millions of dollars. For instance, the development of OpenAI's GPT-3 was reported to have spent more than $5 million [74]. In response to these huge costs, researchers have initiated efforts to distil the expansive mathematical knowledge contained within LLMs into smaller models. Liang et al. [182] took an innovative approach by using GPT-3 as a form of "maths tutor", employing it to generate specialised maths samples that were then used to train a smaller model. The process aimed to transfer the expansive knowledge of GPT-3 to a more cost-efficient model. Yet, it remains an open question as to the extent of mathematical knowledge such "student" models can acquire from their larger counterparts.

This thesis introduces a much smaller model, comprised of roughly 500M parameters contrast to the tens or hundreds of billions of parameters seen in current LLMs. Despite its relatively reduced size, our model demonstrates performance on par with these larger models in mathematical reasoning tasks. This not only underscores the effectiveness of knowledge distillation techniques but also highlights the potential for cost-effective AI models that retain a high level of mathematical reasoning ability.

2.3 Automated Geometry Maths Problem-solving Overview

The previous section provides the review of the realm of text-based automatic numerical reasoning. This section further explores and evaluates effective methods for automated reasoning within the context of geometry maths problems. This section will present a comprehensive review of existing public datasets in section 2.3.1 and the various methods proposed in this research field in section 2.3.2.

2.3.1 Benchmarks for Geometry Mathematical Problems

In contrast to the abundance of datasets for automated mathematical reasoning in text, the realm of geometry mathematics problems faces a notable scarcity of high-quality datasets. Existing datasets for geometry problems are often hindered by limitations such as small scale or lack of public accessibility, presenting a significant challenge in this field of study [43–45, 69].

The growing interest in leveraging deep-learning approaches for solving geometry maths problems has spurred the creation of several public datasets in recent years [11, 60, 61, 183]. A notable one in this area is the introduction of Geometry3K [60], the first large-scale dataset for deep-learning methods in this domain. It offers examples that combine text and diagrams with structured descriptions in a formal language. However, the Geometry3K dataset lacks annotations for the reasoning process, a key element for developing mathematical reasoning skills, as discussed in Section 2.2.1. In response to this limitation, Chen et al. [11] introduced GeoQA, which encompasses 4,998 geometry problems. Each problem in GeoQA is accompanied by a reasoning program expressed in a domain-specific language, enriching the dataset's utility for deep-learning applications.

Earlier datasets were constrained by a limited variety of problem types, predominantly focusing on geometry problems related to angles and lengths. This lack of diversity not only hindered the evaluation of models against more challenging problems but also limited the exploration of potential advantages offered by a wider range of problem types. To address these limitations, Cao et al. [61] expanded the GeoQA dataset by adding an additional 2,518 problems encompassing various types. This expansion resulted in the creation of the GeoQA+ dataset, in a total size of 12,054 problems, thereby significantly enhancing its diversity. More recently, the introduction of UniGeo [1] marked a significant milestone, being the first dataset to include a new category of geometry maths problems—proving problems. UniGeo stands out as the sole benchmark by far that encompasses both geometry calculation and proving problems.

Solving geometry maths problems poses a distinct challenge compared to text-based maths reasoning problems due to the necessity of interpreting geometric diagrams. This additional complexity requires training AI systems to effectively parse these diagrams using extensive data. In response to this specific need, Zhang et al. [5] meticulously annotated geometric diagrams at a primitive level and introduced the PGPS9K dataset. Comprising 9,022 geometry problems each accompanied by a diagram, PGPS9K stands out from existing datasets. It uniquely features both detailed diagram annotations and reasoning programs, providing a comprehensive resource for research in this area. Please refer to Table 2.2 for a complete overview of geometry maths problem datasets.

Name	Diagram Annotation	Reasoning Annotation	Size	Year	Public Available
GeoOS [43]	-	-	186	2015	No
GeoShader [44]	-	-	102	2017	No
GEOS++ [45]	-	-	1,406	2017	No
GEO-OS [46]	Logic Form	Logic Form	2,235	2017	No
Geometry3K [60]	Logic Form	Logic Form	3,002	2021	Yes
GeoQA [11]	-	Solution Program	4,998	2021	Yes
GeoQA+ [61]	-	Solution Program	12,054	2022	Yes
UniGeo [1]	-	Solution Program	14,541	2022	Yes
PGPS9K [5]	Primitive Level Annotation	Solution Program	9,022	2023	Yes

Table 2.2: Datasets for the automated geometry maths problem-solving. $-$ indicates "not available".

2.3.2 Methods for Geometry Maths Problem-solving

A variety of methods have been proposed from the inception to automate the solving of geometry maths problems [184–187]. These problems are distinct from standard maths word problems due to the inclusion of geometric diagrams, requiring a more robust reasoning capability to interpret multi-modal information. From a psychological standpoint, tackling geometry problems demands advanced thinking skills, specifically in symbolic abstraction and logical reasoning [188, 189]. This necessitates the development of specialised methods tailored specifically for addressing the complexities inherent in geometry maths problems.

The current methods for addressing geometry maths problems fall into two primary categories: symbolic solvers [43, 45, 60] and neural solvers [1, 11]. Symbolic solvers rely on semantic parsing, a process that converts geometry diagrams into a formal language using a specific set of rules and a well-defined alphabet. This formal language is then utilised to generate solution programs. Semantic parsing traditionally involves mapping language to a formal meaning representation, initially learned from natural language utterances paired with logical forms [190, 191]. GEOS [43], a pioneering automated system in this field, utilises dual parsers to interpret both texts and diagrams. It correlates diagram components with textual entities and transforms these relations into formal language, subsequently processed by a solver to deduce the final answer. This approach has influenced subsequent research in the area [44–46].

However, Sachan et al. [45] identified a critical limitation in these methods: they cannot incorporate external knowledge for problem-solving. Addressing this gap, they proposed a method that integrates knowledge extracted from textbooks with the solvers, enhancing their alignment with parsed text and diagram information. While symbolic solvers generally surpass neural solvers in performance, mainly due to their precise diagram descriptions in formal language, they are heavily reliant on human-annotated diagram components as intermediate steps. Moreover, they lack explicit reasoning processes in their predictions, reducing the task to an optimisation problem that depends on satisfying parsed information [43–45]. Their reliance on a limited set of handcrafted rules and validation on small-scale datasets limits their generalisability to more complex and real-world geometry maths problems.

Moving away from the labour-intensive process of manually crafting rules, researchers have turned to neural network architectures, known as neural solvers. These solvers work by embedding diagrams and problem texts either separately or jointly, leading to models with a high degree of generalisation and eliminating the necessity for specific rules to discern geometric relations. For instance, neural networks are now capable of supplanting human effort in translating texts and diagrams into formal languages [60]. For embedding diagrams and problem texts, various neural networks are employed. The network designated for diagrams typically specialises in object detection, using either object detection methods like Faster-RCNN [192] and FCOS [8] or instance segmentation methods like Mask R-CNN [193], based on backbone networks such as ResNet [6] or MobileNet [7]. Meanwhile, text inputs are processed using encoders, which could be either traditional networks like LSTM [133] and GRU [134] or more advanced ones like transformers [2]. The key step in this process is the amalgamation of the text and diagram embeddings into a unified representation. Techniques such as BAN [194], FiLM [195], and DAFA [196] are utilised for this purpose. Moreover, integrating auxiliary tasks can further refine these joint representations [11]. These tasks primarily focus on enhancing the semantic representation between text and diagram, and include tasks like diagram geometric element prediction, and knowledge point prediction.

Previous iterations of neural solvers processed text inputs and diagrams independently, which limited their ability to develop effective joint representations. Addressing this, Chen et al. [1] introduced the Geoformer, a novel unified geometric transformer. Built upon VL-T5 [4], which features a bidirectional multi-modal encoder and an auto-regressive text decoder, the Geoformer can uniformly encode multiple inputs. However, it struggles with accurately recognising symbolic characters from text inputs (e.g., $\triangle ABC$), as VL-T5's tokenizer tends to break down these symbols into individual letters. To overcome this issue, Ning et al. [197] developed a method that aligns diagram feature counterparts with the symbolic character embeddings in the text encoder, thereby extracting visual information about these symbols.

While neural solvers have the advantage of not requiring the design of specific rules for identifying geometric relations, they face their own set of challenges. Accurately depicting the intricate relationships among components in geometry diagrams remains a significant hurdle. Additionally, their approach of representing geometric relations as vectors poses a challenge in terms of human interpretability. This lack of clarity makes it difficult to diagnose issues and determine whether performance limitations are due to flaws in the relation extractor or the program solver that generates the reasoning process.

While symbolic solvers excel in precisely depicting components from diagrams, their reliance on extensively crafted rules is a significant drawback. Conversely, neural solvers eliminate the need for such rule design but fall short in providing exact diagram descriptions, particularly for diagrams with slender, overlapping components in complex spatial arrangements. Bridging these gaps, Zhang et al. [5] introduced a method that represents geometric diagrams through structural clauses and semantic clauses achieving a more precise description than traditional symbolic solvers. These clauses are derived using networks based solely on neural nets, thus avoiding the need for complex rule creation characteristic of symbolic solvers.

In this thesis, we present an innovative architecture that translates diagrams into natural language descriptions. This approach not only eliminates the necessity for intricate rule formulation but also enhances extensibility. The use of natural language descriptions not only improves interpretability but also bride the gap between different modalities, such as geometric diagrams and problem texts. Furthermore, this method is designed to integrate seamlessly with LLMs, enabling to utilise the power of advanced LLMs and allowing for its substitution with any transformer-based model. Finally, to drive progress in automated solving geometry maths problems, we establish a benchmark for assessing the capabilities of LLMs and MMs in resolving such challenges. We offer detailed analysis by comparing the most recent advancements in LLMs and MMs, providing a thorough understanding of their proficiency in solving geometry maths problems.

2.4 Chapter Summary

This chapter comprehensively reviews three areas: large language models, text-based automated numerical reasoning, and automated geometry maths problem-solving, all of which are integral to the concepts explored in this thesis.

In our analysis of text-based automatic numerical reasoning, we identify two primary challenges. First, existing methods struggle with complex reasoning, often failing to handle tasks involving diverse operators and dynamic operand counts due to their inability to separate operator and operand generation. This results in compounded errors in more intricate scenarios. Second, there is a notable lack of extensibility in operators, stemming from either model architecture limitations or program representation formats, thereby restricting application across different domains.

Moreover, while there is a significant effort to address geometry maths problems, current methods predominantly rely on LLMs or recurrent neural networks to generate solution programs. These approaches fall short in inherently capturing the unique aspects of geometry maths problems. Additionally, these methods often fail to accurately depict the intricate relationships among diagram components, thus hindering the effective resolution of geometry maths problems.

In response to these challenges, the next chapter introduces a novel architecture designed specifically for text-based numerical reasoning task. This is followed by a deeper exploration into an architecture tailored for geometry maths problem-solving, focusing on improved representation of geometric diagrams and more effective integration of text and diagram descriptions.

Part II

Text-based Automated Numerical Reasoning

In Part II, we delve into a model uniquely designed for the task of text-based automated numerical reasoning. Chapter 3 presents the num**E**rica**L** re**AS**oning with adap**T**ive symbol**I**c **C**ompiler (ELASTIC) model. This model is tailor-made to tackle challenges in this task. The insights and methodology of this chapter are based from my published work, "ELASTIC: Numerical Reasoning with Adaptive Symbolic Compiler" presented at the Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS 2022) in (`https://proceedings.neurips.cc/paper_files/paper/2022/hash/522ef98b1e52f5918e5abc868651175d-Abstract-Conference.html`).

Chapter 3

ELASTIC: Numerical Reasoning with Adaptive Symbolic Compiler

Text-based automated numerical reasoning is a challenging task of Artificial Intelligence (AI), requiring reading comprehension and numerical reasoning abilities. As discussed in Section 2.2, previous approaches use reasoning programs to represent the reasoning process. However, most works do not separate the generation of operators and operands, which are key components of a reasoning program, thus limiting their ability to generate such programs for complicated tasks. The proposed ELASTIC model in this chapter is constituted of the RoBERTa [79] as the Encoder and a Compiler with four modules: Reasoning Manager, Operator Generator, Operands Generator, and Memory Register. ELASTIC is robust when conducting complicated reasoning. Also, it is domain agnostic by supporting the expansion of diverse operators without caring about the number of operands it contains. We conduct extensive experiments to show that ELASTIC outperforms previous state-of-the-art models.

The rest of the chapter is organised as follows: Section 3.1 briefly describes the our work with respect to other state-of-the-art methods proposed for the text-based automated numerical reasoning task. Additionally, we list the research questions of this work and contributions of this work. Section 3.2 describes the definition of the task and the details of our proposed ELASTIC model. Section 3.3 gives the setup details of the experiments of this work. And Section 3.4 shows results of our experiments and give comprehensive analysis of the experimental results according to the research questions proposed in Section 3.1. Finally, Section 3.5 summarises this chapter.

3.1 Introduction

Recently, Pre-trained language models (PLMs) [74, 78, 79, 87, 198] show astonishing performance over reading comprehension tasks like SQuAD [199]. However, PLMs fall short of numerical reasoning over text [58], which requires conducting numerical reasoning based on understanding the text. Hence, text-based numerical reasoning is more challenging than reading comprehension [168] and attracts the interest of the AI community. Previous approaches adopt the sequence-to-sequence architecture to generate the sequential format of numerical reasoning programs (see (b) in Table 1.1) [105]. However, the sequential format could produce invalid expressions such as "$3 - ((2)$" because of the wrong position of parentheses [146]. To avoid this, some methods convert the reasoning program to the binary tree, then use the tree-decoder to generate the pre/post-order traversal sequence (see (c) in Table 1.1) [106,147,200]. Alternatively, FinQANet [10] represents the reasoning program in a flattened format and generates the right parentheses forcibly after generating two consecutive operands. To increase the scalability, NeRd [58] introduces the symbolic operations and generates the reasoning program as the nested compositional format (see (e) in Table 1.1). Researchers also investigate to capture valuable information between entities and numbers to improve numerical reasoning ability. Some works use PLMs [58, 168, 171], while others, like Li et al. [158] and Ran et al. [201], adopt graph neural network to encode the text.

Currently, proposed approaches struggle with two significant problems. Firstly, they are vulnerable to complicated numerical reasoning problems. The complicated numerical reasoning problems usually contain a long reasoning program, in which the types of operators are diverse, and the number of operands is dynamic. Since most works do not separate the generation of operators and operands, their performance is hindered by cascading errors when encountering complicated tasks. Secondly, previous works lack extensibility for the operators, which arises from either the flaw of the model architecture or the representation format of the program, making them hard to apply to different data domains.

Hence, in this chapter, we state that independently generating the operators and operands in reasoning programs can minimise the risk of cascading errors, and further can resolve more complex reasoning programs (See Statement (1) in Section 1.3). Specifically, this chapter presents the num**E**rica**L** re**AS**oning with adap**T**ive symbol**I**c **C**ompiler (ELASTIC) model. ELASTIC separates the generation of operators and operands, allowing it to be less influenced by the cascading error from the complicated reasoning. Moreover, ELASTIC is adaptable to the number of operands following an operator, making it domain-agnostic to support diverse operators. Specifically, ELASTIC contains an Encoder part extracting the contextual representations of the passage and question and a Compiler part generating the numerical reasoning program. The Compiler consists of four modules: Reasoning Manager, Operator Generator, Operands Generator, and Memory Register.

To validate our thesis statement, we conduct experiments to answer the following research questions (RQ):

- **RQ-3.1 Performance Evaluation**: To what extent does the ELASTIC model outperform the existing state-of-the-art models in execution accuracy and program accuracy on the FinQA and MathQA datasets?
- **RQ-3.2 Effectiveness of Separation**: How does the separation of the generation of operators and operands impact the effectiveness of generating reasoning programs in dealing with complicated tasks?
- **RQ-3.3 Generalisability and Adaptation**: How well can the ELASTIC model generalise across different domains, and how efficiently can it incorporate diverse operators without being affected by the number of operands they contain?
- **RQ-3.4 Robustness Analysis**: How does the ELASTIC model maintain robustness when conducting complicated reasoning tasks, which is reflected by the

length of the program steps, and what are the challenges encountered during these tasks?

- **RQ-3.5 Impact of Memory Register**: What is the individual contribution of the Memory Register in the overall performance of the ELASTIC model to utilise the results from the previous sub-programs?

3.1.1 Contributions

The contributions of this chapter are:

1. This chapter presents the ELASTIC model with good adaptability and elasticity, which separates the generation of operators and operands. ELASTIC achieves state-of-the-art results on two challenging datasets: FinQA and MathQA.
2. This chapter introduces the design of separate modules and Memory Register, making ELASTIC perform stably on complicated reasoning problems.
3. The proposed ELASTIC is domain agnostic because it supports diverse operators.

3.2 Approach

This section describes the detail of the ELASTIC model. Figure 3.1 shows the architecture of our ELASTIC model. ELASTIC consists of an Encoder part encoding the question text and problem text into contextual vectors and a Compiler part producing the numerical reasoning programs. The Compiler part consists of four modules: Reasoning Manager, Operator Generator, Operands Generator, and Memory Register. The Reasoning Manager leverages other modules to produce the numerical reasoning program. Since a complete numerical reasoning program usually contains several sub-programs, the generation steps between operators and operands are interchangeable. To help the following sub-programs use executable results from the previous sub-programs, Memory Register stores the sub-programs executable results into corresponding pre-defined cache tokens embeddings.

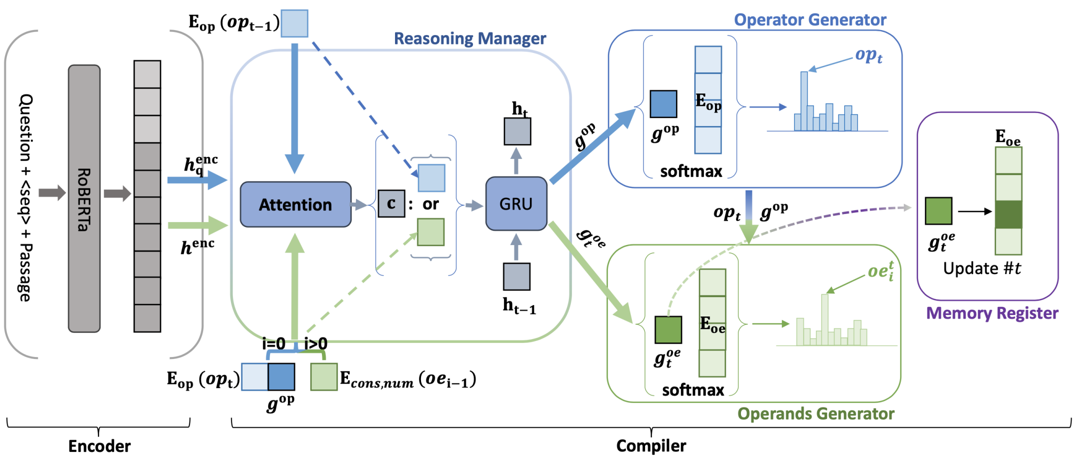


Figure 3.1: The overall architecture of the ELASTIC model. The Encoder part takes the sequence of question text Q and passage text P as input, then generates the contextual vectors $\mathbf{h}^{enc}$. The Compiler part consists of four modules: **Reasoning Manager**, **Operator Generator**, **Operands Generator**, and **Memory Register**. The right part of the figure shows a complete process of the generation of sub-program r_t. Firstly, Reasoning Manager sends the guidance vectors g^{op} to the Operator Generator, which guides the generation of operator op_t. Secondly, Reasoning Manager suspends the Operator Generator, then the Operands Generator takes g^{op} and op_t from the Operator Generator to produce the first operand oe_1^t. When finish the generation of the sub-program r_t, the Memory Register stores the results and updates the embedding vectors of cache token $\#t$ by g_t^{oe}. Again, the Compiler repeats to generate next sub-program r_{t+1}.

3.2.1 Task Definition

Given the problem text P and question text Q, the task is to generate a numerical reasoning program R. Both problem text P and question text Q consist of words and numbers (denoted by NUM). The numerical reasoning program R represents the numerical reasoning process, which is a sequence of symbols (denoted by s) from mathematical operators (denoted by OP) and operands (denoted by OE). Operands OE are from either constant numbers (denoted by $CONS$) defined in Domain Specific Language (DSL) or NUM. $CONS$ are the special numbers that do not exist in either the problem text P and question text Q, such as const_pi (π). Finally, the pattern of the numerical reasoning program R is defined as $R = \left\{ op_i \left[oe^i_j \right]_{j=0}^{m-1} \right\}_{i=0}^{n-1}$, where $op_i \in OP$, it is the i_{th} operator in R, and op_i contains several operands oe^i_j. In addition, we regard a group of one operator and its operands as the sub-program r. For example, $op_i \left[oe^i_j \right]$ is the i_{th} sub-program r_i, which can be executed since it is a complete arithmetic program. See Table 3.1 for the definition of all the notations. Please see Appendix A.1 for an example of a maths problem and its reasoning programs.

Notation	Description
P, Q, R	Problem Text, Question Text, Numerical Reasoning Program
NUM	The numbers in P and Q
$CONS$	Constants defined in DSL
OP	All mathematical operators
op_i	The ith operator in R
OE	All operands
oe^i	All operands belonging to op_i
oe^i_j	The jth operands of op_i
s	From either OP or OE, s constitute R
r_i	The i-th sub-program of R $r_i = op_i \left[oe^i \right]$

Table 3.1: Task Definition Notation.

3.2.2 Encoder Part

As shown in Figure 3.1 (Encoder), the Encoder takes the concatenated sequence of Q and P as input. The Encoder encodes the input sequence and outputs the contextual vectors $\mathbf{h}^{enc}$. Next, $\mathbf{h}^{enc}$ is used for the Compiler to produce the numerical reasoning program R. In this work, we use RoBERTa [79] as the Encoder. The outputs from the final layer of RoBERTa is used as $\mathbf{h}^{enc} \in \mathbb{R}^{h*s}$, where s is the maximum input length of the RoBERTa, and h is the hidden size of RoBERTa. Note that ELASTIC is not dependent on the specific type of encoder. Any model providing contextual vectors of the sequence can be used.

3.2.3 Compiler Part

3.2.3.1 Decoding Vocabulary and Token Embedding

We first describe the decoding vocabulary. The decoding vocabulary consists of *OP* and *OE*, where *OE* can be further categorised into *NUM* and *CONS*. The embedding $\mathbf{e}_s$ of symbol s of the decoding vocabulary is represented by the embedding $\mathbf{E}_{op,cons,num}(s)$, which is the embedding look-up function. Hence, the embedding for symbol s is defined as:

$$\mathbf{e}_s = \begin{cases} \mathbf{E}_{op}(\text{s}) & \text{if s} \in \mathit{OP} \\ \mathbf{E}_{cons}(\text{s}) & \text{if s} \in \mathit{CONS} \\ \mathbf{E}_{num}(\text{s}) = \mathbf{h}_i^{enc} & \text{if s} \in \mathit{NUM} \end{cases} \tag{3.1}$$

The symbols embeddings of *OP* and *CONS* are two trainable embedding matrices $\mathbf{E}_{op} \in \mathbb{R}^{h*n_{op}}$ and $\mathbf{E}_{cons} \in \mathbb{R}^{h*n_{cons}}$ (n_{op} and n_{cons} refers sizes of *OP* and *CONS* respectively). The embedding for the symbol of *NUM* is $\mathbf{h}_i^{enc} \in \mathbb{R}^h$, where i denotes the index position in the sequence of $\mathbf{Q}$ and $\mathbf{P}$.

3.2.3.2 Reasoning Manager

As shown in Figure 3.1 (Reasoning Manager), the Reasoning Manager outputs the vector $\mathbf{g}$, which guides the Operator Generator and the Operands Generator to produce *op* and *oe*. The inputs for the Reasoning Manager are contextual vectors $\mathbf{h}^{enc}$ ($\mathbf{h}_q^{enc}$ for generating operators) from the Encoder and embedding of the previously generated symbol s_{t-1}. The Reasoning Manager first calculates the context vector $\mathbf{c}$ by the normalised vectors of $\mathbf{h}_i^{enc}$ and the attention weights a_i:

$$\mathbf{c} = \sum_i a_i \mathbf{h}_i^{enc} \tag{3.2}$$

$$a_i = \frac{\exp(\text{score}(\mathbf{e}_{s_{t-1}}, \mathbf{h}_i^{enc}))}{\sum_j \exp(\text{score}(\mathbf{e}_{s_{t-1}}, \mathbf{h}_j^{enc}))} \tag{3.3}$$

$$\text{score}(\mathbf{e}_{s_{t-1}}, \mathbf{h}_i^{enc}) = \mathbf{e}_{s_{t-1}}^{\mathrm{T}} \mathbf{W}_1 \cdot \mathbf{W}_2 \mathbf{h}_i^{enc} \tag{3.4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{h*h}$ and $\mathbf{W}_2 \in \mathbb{R}^{h*h}$, and both are trainable parameters. The $\mathbf{c}$ summarises the encoded information from the Encoder according to the previous generated symbol s. Next, the Reasoning Manager adopts the GRU [134] network to generate the guidance output $\mathbf{g}$:

$$\mathbf{g}, \mathbf{h}_t = \text{GRU}(\text{ReLU}(\mathbf{W}_3[\mathbf{c} : \mathbf{E}_{op,cons,num}(s_{t-1})]), \mathbf{h}_{t-1}) \tag{3.5}$$

where ":" represents concatenation. $\mathbf{W}_3 \in \mathbb{R}^{h*2h}$ is trainable parameter, and ReLU is the activation function. $\mathbf{h_{t-1}} \in \mathbb{R}^h$ is the hidden state of GRU from the previous step, and $\mathbf{h}_0$ is $\mathbf{0}$.

3.2.3.3 Operator Generator

As shown in Figure 3.1 (Operator Generator). Firstly, the Operator Generator receives the guidance vector $\mathbf{g}_t^{op}$ from the Reasoning Manager by inputting: contextual vectors $\mathbf{h}_q^{enc}$ of tokens from the question Q, and embedding $\mathbf{E}_{op}(op_{t-1})$ of the previously generated operator. Next, the Operator Generator calculates the probabilities of i-th operator (denoted as i-op) of the OP:

$$\mathbb{P}(i\text{-}op|\mathbf{E}_{op}(op_{t-1}), \mathbf{g}_t^{op}) = \frac{\exp(\mathbf{E}_{op}^{\mathrm{T}}(i\text{-}op)\text{ReLU}(\mathbf{W}_{op}\mathbf{g}_t^{op}))}{\sum_{j\text{-}op \in OP} \exp(\mathbf{E}_{op}^{\mathrm{T}}(j\text{-}op)\text{ReLU}(\mathbf{W}_{op}\mathbf{g}_t^{op})} \tag{3.6}$$

where $W_{op} \in \mathbb{R}^{h*h}$ is trainable parameter. The Operator Generator selects the operator with the highest probability as the predicted *op*. Next, unlike other models, the Reasoning Manager suspends the generation of operators and starts to generate operands $\{oe^t\}$ through the Operands Generator.

3.2.3.4 Operands Generator

As shown in Figure 3.1 (Operands Generator). The inputs from Operands Generator to the Reasoning Manager are different from Operator Generator's. Because *oe* could be a number in Q or P, the contextual vectors $\mathbf{h}^{enc}$ of all tokens are used. Furthermore, the Operands Generator initialises the embedding of the initial operand $\mathbf{e}_{(oe_0^t)}$ as $\text{ReLU}(\mathbf{W}_4[\mathbf{E}_{op}(op_t) : \mathbf{g}_t])$ ($\mathbf{W}_4 \in \mathbb{R}^{h*2h}$), leveraging information of op_t to produce oe^t. Next, the Reasoning Manager outputs g_n^{oe} for n-th step generation of operand oe_n^t. Finally, the probability of i-th operand (denoted as *i-oe*) of the OE:

$$\mathbb{P}(i\text{-}oe|\mathbf{E}_{cons,num}(oe_{n-1}^t), \mathbf{g}_t^{oe}) = \frac{\exp(\mathbf{E}_{cons,num}^{\mathrm{T}}(i\text{-}oe)\text{ReLU}(\mathbf{W}_{oe}\mathbf{g}_t^{oe}))}{\sum_{j\text{-}oe\in OE}\exp(\mathbf{E}_{cons,num}^{\mathrm{T}}(j\text{-}oe)\text{ReLU}(\mathbf{W}_{oe}\mathbf{g}_t^{oe})} \tag{3.7}$$

where $W_{oe} \in \mathbb{R}^{h*h}$ is trainable parameter. The Operands Generator selects the operand with the highest probability as the predicted *oe*. After one operand has been generated, the Operands Generator continues producing operands for the sub-program r_t. The decoding process for the operands terminates when the token *none* is produced.

3.2.3.5 Memory Register

When generating sub-program r_i, its operands could be the executable results from the previous sub-program $r_p (p < i)$. To make the Operands Generator be able to use the results from previous sub-programs. Inspired by Chen et al. [10], we introduce a cache token $\#n$ to the *CONS* of DSL, which is used for storing the information of executable results. Unlike other constants, $\#n$ does not point to a static value. It is different according to the different sub-program r_n. As the results, ELASTIC needs to update the representation of $\#n$ after the sub-program r_n is generated. Specifically, the Memory and Register module update the cache $\#n$ by replacing its embedding with output $\mathbf{g}_n^{oe}$, which is the guidance vector from Reasoning Manager to guide the generation of the last operands belonging to the sub-program r_n.[1]

3.2.4 Training Objective

Given the data $\mathbb{D}$ with size of N containing $P_i, Q_i, \hat{op}^i, \hat{oe}^i$, where P_i and Q_i refer to the passage and question in the the i^{th} training data, likewise, $\hat{op}^i$ and $\hat{oe}^i$ are the golden operators and operands. Our training goal is to minimise the sum of the negative log-likelihood over the entire data, so the training loss is as follow:

$$\text{Loss} = \sum_{i=1}^{N} -\left\{\log \mathbb{P}(\text{OP}^i|\text{P}^i, \text{Q}^i) + \log \mathbb{P}(\text{OE}^i|\text{P}^i, \text{Q}^i)\right\} \tag{3.8}$$

3.3 Experimental Setup

3.3.1 Datasets

In this section, we conduct evaluation experiments on two datasets: FinQA [10] and MathQA [9].[2]

- **FinQA:** FinQA is a dataset created from the annual financial reports. It contains 8,281 data, split into train, eval, and test parts with 6,251, 883, and 1,147 examples. We adopt the evaluation metrics from the original FinQA paper: execution accuracy (Exe Acc) and program accuracy (Prog Acc). The program accuracy calculates the accuracy of the operators and operands between the predicted program and the golden program. The execution accuracy calculates the accuracy between the golden executable result and the result from the predicted program. Since the FinQA dataset only contains operators with two operands, we extend it by creating questions required to be solved by the operators with

[1]Please see Appendix A.2 for an example to show how separated modules work.

[2]We do not select other datasets because of: (1) too small in size (around 1000), e.g., MAWPS [112] and ASDiv-a [116], (2) language are not English. e.g. Math23K [105] and HMWP [106], (3) lack intermediate annotated program, like DROP [59].

more than two operands. We use the extended FinQA dataset to evaluate our models' adaptability to the number of operands (See Section **??**).

- **MathQA:** MathQA is created from GRE/GMAT examinations, containing 37,200 maths word problems. The dataset is split into 80%, 12%, and 8% of train, dev, and test data. Compared with the FinQA dataset, the examples of MathQA require more advanced reasoning ability, which challenges the model to conduct advanced numerical reasoning.[3] A significant difference with FinQA is that the number of operands following an operator is not explicit in the MathQA dataset. Each MathQA question contains one correct of several answer options, calculated by the reasoning program with the knowledge of the operation semantics. Since we do not have this kind of knowledge, we adopted the same way as NeRd [58], by only using program accuracy to evaluate models' performances. Note that program accuracy is stricter than execution accuracy because the model could find the correct answer by spurious reasoning programs.

3.3.2 Baselines

We compare our ELASTIC model with several state-of-the-art models. (1) **FinQANet** [10]: It adopts the Encoder-Decoder architecture with a cache updating mechanism to generate the program. Since FinQANet only supports generating operators with exact two operands, we manage to train and evaluate FinQANet on the MathQA dataset by discarding the operators containing more than two operands. (2) **NeRd** [58]: it uses the BERT and a pointer-generator-based model to generate the symbolic nested program. (3) **Graph2Tree** [158]: It models the dependency information of the text sequence by the GraphSAGE [202] like model, and generates the program in a tree-structured way. (4) **NumNet** [201]: NumNet models the numeracy information by a GNN network. We also train the **NumNet+**, which replaces the Encoder of the NumNet by RoBERTa-large.[4] Note that program accuracy does not apply to NumNet, since NumNet does not generate compositional reasoning programs. (5) **Human Performance**: We also

[3]Please see comparison between two datasets in Appendix A.3.

[4]https://github.com/llamazing/numnet_plus

report the human performance of both experts and non-experts in the FinQA dataset. The results are taken from the original FinQA paper [10].

3.3.3 Implementation Details

The model is implemented by PyTorch [203] and Transformers [204], then trained on one NVIDIA A100 GPU (40G). Training epochs are set to 50 and 100 for FinQA and MathQA, respectively. The batch size for all datasets is set to 10. We use Adam as optimiser [205] to update the parameters of the models. The initial learning rate is set to 1e-5 equally, and it would be halved in every 25 epochs and 50 epochs for FinQA and MathQA. During training, the dropout rate and the weight decay are set to 0.1 and 1e-5 to prevent over-fitting. The parameters of the RoBERTa are fine-tuned during training. For the GRU cell in the decoder, the hidden size is the same as the RoBERTa, and the GRU layers number is 4. During inference, we use greedy decoding to generate the reasoning program.

3.4 Experimental Results and Discussions

3.4.1 Comparison with Previous Methods (RQ-3.1, RQ-3.2, RQ-3.3)

Table 3.2 shows the performances of our ELASTIC model and baselines on FinQA and MathQA datasets. Overall, ELASTIC (RoBERTa-large) achieves the highest scores on both datasets. In the FinQA dataset, we see a lead in our ELASTIC (RoBERTa-large) model compared to the best baseline FinQANet (RoBERTa-large), with 3.91 points higher execution accuracy and 1.69 points higher program accuracy. When we change the Encoder part of ELASTIC from RoBERTa-large to RoBERTa-base, it still achieves better results than FinQANet using the same size of RoBERTa. Since both ELASTIC and FinQANet use the RoBERTa as the encoder, the results demonstrate the improvements brought by separating the generation procedures for operators and operands. Both ELASTIC models outperform the NeRd with a large margin. It is worth mentioning that NeRd defines external rules for different operators in their model [58], which is not the case with the ELASTIC. ELASTIC also outperforms the NumNet

Datasets & Metrics	FinQA (test)		MathQA (test)
	Exe Acc	Prog Acc	Prog Acc
Graph2Tree	0.37	0.0	69.96†
NumNet	2.32	n/a⋆	n/a⋆
NumNet+	10.29	n/a⋆	n/a⋆
NeRd	52.48‡	49.90‡	79.70†
FinQANet (RoBERTa-base)	60.10†	58.38†	74.12
FinQANet (RoBERTa-large)	65.05†	63.52†	79.20
ELASTIC (RoBERTa-base)	62.66	59.28	82.27
ELASTIC (RoBERTa-large)	**68.96**	**65.21**	**83.00**
Human Expert	91.16†	87.49†	n/a
Human Non-Expert	50.68†	48.17†	n/a

Table 3.2: Overall Results for the baselines and ELASTIC on the testing data from three datasets. † means that the scores are taken from the original papers. ‡ means that the scores are taken from the FinQA paper [10]. ⋆ The program accuracy does not apply to the NumNet on FinQA and MathQA datasets because NumNet does not generate the intermediate reasoning program. In addition, NumNet could only solve reasoning program involving add and subtract operations. However, the proportions of examples only use add and subtract as operations in MahtQA are 0.055% and 0.056%, respectively. As a result, we choose not to train NumNet on MathQA.

and NumNet+ by a considerable margin. This could be due to the internal structure of these models limiting their scalability in generating reasoning programs, thus struggling to produce reasoning steps in a systematic manner [206]. Finally, Graph2Tree achieves only 0.37 accuracy on the FinQA test dataset, which is much lower compared to its 69.96 program accuracy on the MathQA dataset. We suspect that this is because of the data leak problem existing in FinQA train and eval data, where the design of Graph2Tree is vulnerable to. Although ELASTIC surpasses the non-expert performance, we can still find a large gap between our ELASTIC model and Human Expert.

For the MathQA dataset, ELASTIC (RoBERTa-large) is the best performing model, with 3.3 higher program accuracy than NeRd and 13.04 points higher than Graph2Tree. We further investigate the performance of ELASTIC using RoBERTa-base, which still achieves higher accuracy of 82.27 than 79.7 of NeRd. The slight performance difference between ELASTIC (RoBERTa-large) and ELASTIC (RoBERTa-base) suggests that the extracted contextual semantic information of passage and question is sufficient. Finally, FinQANet achieves promising results on MathQA, 79.20% program accuracy by FinQANet (RoBERTa-large) and 74.12% program accuracy by FinQANet (RoBERTa-base). Note that we discarded the data of MathQA containing more than two operands for the FinQANet, so that performance of FinQANet on MathQA is the overestimation of its numerical reasoning ability. Even with such consideration, ELASTIC outperforms FinQANet, demonstrating that ELASTIC is more adaptable by supporting diverse operators than FinQANet.[5]

3.4.2 Performance on Different Program Steps (RQ-3.4)

When producing the long numerical reasoning program, ELASTIC is less influenced by the cascading error. To demonstrate this superiority of ELASTIC, we investigate how different lengths of programs influence models' performances.

Table 3.3 displays the models' performances when generating programs with different steps. ELASTIC (RoBERTa-large) performs better than FinQANet (RoBERTa-large)† when the program step is either 1 or 2, indicating ELASTIC also performs well

[5]Please see Appendix A.4 for models' performances on extended FinQA dataset.

Program Steps	ELASTIC		FinQANet†		FinQANet‡		# Train & Dev
	Exe Acc	Prog Acc	Exe Acc	Prog Acc	Exe Acc	Prog Acc	
=1	76.30	75.66	70.27	68.77	73.70	71.25	4240
=2	66.01	66.01	63.69	61.79	62.34	59.65	2300
≥3	31.78	31.10	31.65	31.65	28.57	23.80	594

Table 3.3: ELASTIC and FinQANet performances on FinQA dataset in terms of different program steps. The "# Train & Dev" is the number of training and development data. All models use the RoBERTa-large as the encoder. † means the results of that model are taken from the original FinQA [10] paper. ‡ means that the FinQANet (RoBERTa-large) is re-trained by ourselves.

on the shorter program steps. Surprisingly, with the program step increasing from 3, the accuracy for both ELASTIC and FinQANet tumbles by half compared with the performances on program steps equal 2. We suspect that the FinQA dataset lacks sufficient training examples for the data with more than 3 program steps. Table 3.3 shows that the number of training data with more than 2 program steps is 594 compared to the numbers of data available for program steps equal to 1 (4240) or 2 (2300). For a fair comparison, we retrained the FinQANet (RoBERTa-large) on the FinQA dataset, but ELASTIC still outperforms it in execution accuracy and program accuracy.

From Figure 3.2, ELASTIC (RoBERTa-large) surpasses the FinQANet (RoBERTa-large) on MathQA dataset almost on every program step. Meanwhile, although MathQA is challenging, ELASTIC (RoBERTa-large) still achieves program accuracy over 80.0 when program steps are less than or equal to 8. The model's performance drops when the program steps are equal to 9 and 10, but starts to soar when the program steps are bigger than 10. This demonstrates that ELASTIC performs well when generating longer program steps. As shown in Figure 3.2, we plot the accuracy for the operator generation, which ignores the correctness of operands generation. We could find that the operation accuracy is always higher than the program accuracy regarding different program steps (except for program steps equal to 12). This finding demonstrates the advantage of separating the generation procedure for operators and operands. This finding also reveals that the wrong predictions are because ELASTIC selects the wrong operands. We suspect this is due to too much noise from the context.

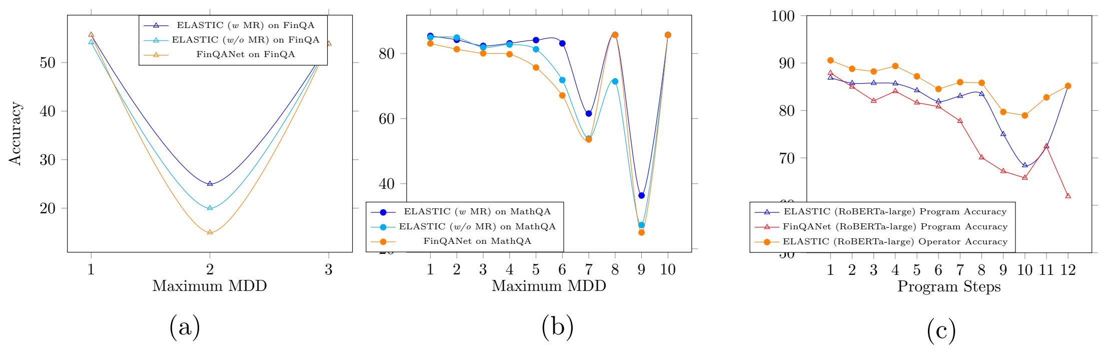


Figure 3.2: (a) Program Accuracy on FinQA according to the M-MDD. (b) Program Accuracy on MathQA according to the M-MDD. (c) Program Accuracy and Operator Accuracy of ELASTIC (RoBERTa-large) on different program steps in MathQA dataset, compared with Program Accuracy of FinQANet (RoBERTa-large).

3.4.3 Necessity of Memory Register (RQ-3.5)

As discussed in the Section Memory Register, ELASTIC stores the executable results of each sub-program into a special cache token #n, and updates its embedding after n-th sub-program is generated. The longer the reasoning program is, the higher the probability of the generating process using the previous sub-program result. This section investigates the effect of the Memory Register on improving numerical reasoning performance.

Datasets & Metrics	FinQA (test)		MathQA (test)
	Exe Acc	Prog Acc	Prog Acc
ELASTIC *w* MR	**68.96**	**65.21**	**83.00**
ELASTIC *w/o* MR	68.79	64.78	82.68
FinQANet	65.06	63.52	79.20

Table 3.4: The performances of ELASTIC with or without memory register (MR). ELASTIC with MR performs better than without MR on FinQA and MathQA datasets. Both ELASTIC with or without MR performs better than FinQANet. All models use the RoBERTa-large as the encoder.

First, we present an ablation study of using and not using the Memory Register for ELASTIC. From Table 3.4, we find that ELASTIC with Memory Register performs slightly better than it without. Similar observations can be found for the MathQA dataset. This observation demonstrates the value of the Memory Register. Next, since ELASTIC and FinQANet store the executable results from the previous sub-program in a different way, we conduct a comparison between the two models. The results from Table 3.4 show that the ELASTIC with Memory Register achieves higher scores than FinQANet on both datasets.

Next, given two sub-program belonging to the same R: r_i, r_j $(i < j)$. where the executable result of r_i is used as the operand for r_j. Then, we introduce the Memory Departing Distance (MDD) for r_i and r_j as $j-i$, and the maximum Memory Departing Distance (M-MDD) as the longest MDD between all $\{r\}$ of R.[6] The bigger M-MDD is, the more challenging to select the correct previous sub-program result, since the model tends to forget the information passing from long steps before. As the result, we investigate how models perform when dealing with different M-MDD.

From Figure 3.2a and Figure 3.2b, the ELASTIC with Memory Register performs better than ELASTIC without it at each M-MDD on FinQA and MathQA datasets. Particularly in the MathQA dataset, when M-MDD is larger than 5, ELASTIC with Memory Register can achieve better results than the ELASTIC without it. This demonstrates the importance of the Memory Register when using executable results from long steps before. Worth mentioning that ELASTIC performs better than FinQANet on both datasets, even without the Memory Register.

3.5 Chapter Summary

In this chapter, we identified two major challenges in handling complex text-based numerical reasoning tasks with existing methods. First, these methods struggle with long reasoning programs characterized by diverse operators and dynamic operand numbers, leading to cascading errors due to their inability to separate operator and operand gen-

[6]For example, in flatten program "$\text{add}(20, 3), \text{subtract}(6, 1), \text{add}(\#1, 10), \text{subtract}(\#0, \#2)$", the MDD for r_0, r_1, and r_2 are 3, 1, and 1. Obviously, the maximum M-MDD is 3.

eration. Second, they exhibit limited extensibility regarding operators, due to either model architecture limitations or the program representation format, hindering their application across different domains.

To address these challenges, this chapter states that independently generating the operators and operands in reasoning programs can minimise the risk of cascading errors, and further can resolve more complex reasoning program. Consequently, this chapter presented the num**E**rica**L** re**AS**oning with adap**T**ive symbol**I**c **C**ompiler (ELASTIC) model. ELASTIC independently generates operators and operands in reasoning programs, minimizing the risk of cascading errors and enabling the resolution of more complex reasoning tasks. Additionally, ELASTIC's domain-agnostic design supports diverse operators, enhancing its adaptability. The model includes a Memory Register to improve performance by utilizing executable results from preceding sub-programs.

To validate the statement, we addressed the research questions outlined in Section 3.1 through extensive experiments comparing ELASTIC with leading methods using two datasets specifically designed for assessing automated mathematical reasoning abilities (see Section 3.3).

We began by evaluating ELASTIC against previous methods on two benchmarks, FinQA and MathQA. This addressed research question **RQ-3.1**, demonstrating that ELASTIC outperforms other methods, achieving the highest scores on both datasets (see Table 3.2).

Next, we conducted a detailed comparison with FinQANet, which utilizes the same encoder as ELASTIC but showed inferior performance. This comparison, discussed in Section 3.4.1, underscores the efficacy of our method's separation design, addressing research question **RQ-3.2**. Additionally, we focused on operator generation accuracy, independent of operand correctness. As illustrated in Figure 3.2, operator accuracy consistently surpasses overall program accuracy across different program steps, highlighting the benefits of separating the generation processes to disentangle errors from operators and operands.

Furthermore, we addressed research question **RQ-3.3** by comparing ELASTIC with other methods across two distinct datasets: FinQA (financial domain) and MathQA (elementary maths word problems). Our results, detailed in Table 3.2, demonstrate that ELASTIC surpasses other methods, showcasing robust performance even with datasets from different domains.

To examine ELASTIC's robustness, addressing research question **RQ-3.4**, we explored the impact of solution program length on model performance in Section 3.4.2. Our findings show that ELASTIC consistently outperforms FinQANet across various program lengths, indicating greater resistance to cascading errors in longer solution programs.

To enhance ELASTIC's performance with lengthy solution programs, we incorporated the Memory Register (MR), detailed in Section 3.2.3.5. Through an ablation study, we addressed research question **RQ-3.5**. The results, shown in Table 3.4, indicate that ELASTIC with MR outperforms its counterpart without MR. Additionally, we introduced the Memory Departing Distance (MDD) and maximum Memory Departing Distance (M-MDD) in Section 3.4.3 to evaluate ELASTIC's efficacy in utilizing results from earlier sub-program steps. The findings demonstrate that ELASTIC with MR yields better outcomes when referencing results from more distant steps compared to ELASTIC without MR.

In the future, we plan to improve the accuracy of matching numbers and entities within the text. Additionally, since ELASTIC requires annotated reasoning programs, which is labor-intensive, we aim to investigate methods for generating reasoning programs from the trained model.

Part III

Automated Geometry Maths Problem-Solving

In the previous part, we introduce a model specifically designed for text-based automated numerical reasoning task. Our extensive experiments have shown that this model not only achieves remarkable performance but also exhibits strong generalisation capabilities. However, the application of mathematical reasoning extends beyond textual data, which exists in multi-modal data as well. Consequently, this part of the thesis focus to the task of solving geometry maths problems, which uniquely incorporate geometric diagrams as part of their input.

Specifically, in Chapter 4, we propose the GeoEval benchmark to prompt the research on automated geometry maths problem-solving task. Subsequently, in Chapter 5, we address the problems of existing approaches in neglecting the distinct characteristics of geometry maths problems by introducing the Geometry-Aware Problem Solver (GAPS) model. Then, in Chapter 6, we tackle the challenge of accurately depicting geometric relations in diagrams. This is achieved through the use of natural language descriptions, which not only improve the interpretability of these relations but also strengthen the connection between different modalities (image and text). The efficacy of these two novel methods is validated through their superior performance across multiple benchmark tests. Finally, in Chapter 7, we investigate the current large language models' and multi-modal models' performances on GeoEval benchmark, providing valuable insights to the community.

Chapter 4

GeoEval Benchmark

This chapter concludes that the datasets currently used for automated geometry problem-solving lack a standardized format and sufficient diversity, which complicates the evaluation of models' true proficiency in solving geometry problems. Additionally, these datasets usually focus on a single type of geometry problem, such as plane geometry, while neglecting other important areas like solid geometry. This omission restricts the ability to perform a comprehensive assessment across the entire range of geometry problems.

To prompt the research, we introduce the GeoEval benchmark in this chapter, a comprehensive collection that includes a main subset of 2000 problems, a 750 problem subset focusing on backward reasoning, an augmented subset of 2000 problems, and a hard subset of 300 problems. This benchmark facilitates a deeper investigation into the performance of models in solving geometry maths problems. This chapter is mainly based on my accepted paper "GeoEval: Benchmark for Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving" in ACL 2024 Findings.

This chapter is organised as follows: Section 4.1 introduces our GeoEval benchmark and summarises the research questions addressed and the contributions of this chapter. Section 4.2 delivers an in-depth presentation of the GeoEval benchmark, detailing its development process and statistical analysis. The chapter concludes with Section 4.3, which offers a recap of the findings and contributions of this chapter.

4.1 Introduction

Geometry maths problems are a key component in assessing the mathematical reasoning skills of K12 students, serving as a critical benchmark for evaluating educational outcomes [207]. The complexity of solving these problems stems from the requirement to interpret both textual and visual information, in addition to applying mathematical reasoning skills. This complexity has made geometry problem-solving a key area of interest for researchers aiming to evaluate the capabilities of AI models in this domain [208–212].

In recent years, several datasets, such as Geometry3K [213], PGPS9K [5], and GeomVerse [214], have been developed to test the proficiency of AI models in solving geometry maths problems. Yet, these datasets often lack a standardised format and sufficient diversity, complicating the assessment of models' genuine proficiency in geometry problem-solving. Furthermore, these datasets typically focus on one type of geometry problem, such as flat geometry, overlooking other crucial areas like solid geometry. This oversight limits the ability to conduct a thorough evaluation across the full spectrum of geometry problems.

To prompt research towards assessing models' proficiency in automated geometry problem-solving, we introduce the GeoEval benchmark, a comprehensive collection specifically designed for this task. GeoEval features its *Comprehensive Variety*, sourced from seven public datasets and formatted uniformly to encompass a wide range of geometric shapes. It includes *Varied Problems*, covering flat, solid, and analytic geometry to challenge models comprehensively. GeoEval supports *Dual Inputs*, accommodating both diagrammatic and textual problem representations. To counter the potential overfitting to previously seen datasets, GeoEval introduces *Diverse Challenges* through backward reasoning, augmented, and hard subsets, each designed to test different aspects of models' geometry problem-solving abilities. Additionally, GeoEval is annotated with *Complexity Ratings*, allowing for a fine-grained analysis of model performance across various difficulty levels, thus providing a robust framework for advancing AI capabilities in understanding and solving geometry maths problems.

4.1.1 Contributions

To summarise, the contributions of this chapter are:

1. We present GeoEval, a benchmark developed to assess the geometry problem-solving capabilities AI models. GeoEval comprises four distinct subsets, each designed to facilitate a thorough evaluation.

4.2 GeoEval Dataset

Total Numbers	
- GeoEval-2000	2,000
- GeoEval-backward	750
- GeoEval-aug	2,000
- GeoEval-hard	300
Input Types	
- text + description	1,120
- text + diagram	1,120
- text + description + diagram	1,166
Answer Types	
- number	5,050
- expression	232
- coordinate	68
Problem Types	
- flat geometry	5,050
- solid geometry	272
- analytic geometry	28
Others	
- average problem length	28
- average description length	34
- geometry shapes	12

Table 4.1: Statistics of GeoEval benchmark.

The GeoEval benchmark is structured into four subsets: GeoEval-2000, comprising 2,000 problems; GeoEval-backward, with 750 problems; GeoEval-aug, containing 2,000 problems; and GeoEval-hard, including 300 problems. Table 4.1 presents a statistical breakdown of the GeoEval benchmark. This benchmark encompasses a total of 5,050 geometry math problems, categorised into four subsets: GeoEval-2000 (2,000 problems), GeoEval-backward (750 problems), GeoEval-aug (2,000 problems), and GeoEval-hard (300 problems). Besides the problem text, each problem in the dataset includes at least one of the following: a geometric diagram, a description of the diagram, or both. The majority of the correct answers are numerical values, with a minority comprising expressions, coordinates, or option letters, primary in the GeoEval-hard subset.

The subsequent sections will detail the collection process for each individual subset, followed by an explanation of the unique features of the GeoEval benchmark.

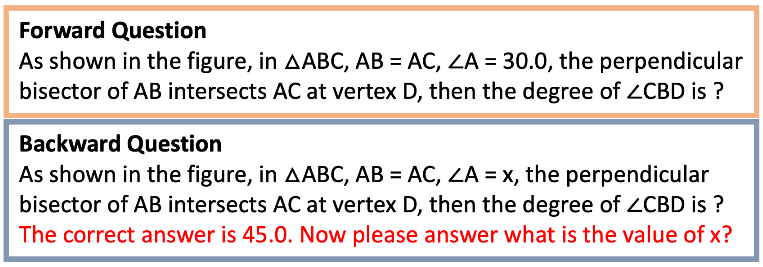
4.2.1 Data Collection

4.2.1.1 Collection from Diverse Data Sources

We have compiled a comprehensive collection of public geometry maths problem datasets, with a total of 24,912 geometry maths problems from sources such as Geometry3K [213], PGPS9K [5], UniGeo [1], GeoQA+ [61], GeometryQA [215], as well as geometry problems from the MATH [216] and MathQA [9] datasets. The first four datasets feature geometry questions that include both problem texts and geometric diagrams, whereas the latter three datasets comprise questions that only contain problem texts.

Building on the data gathered, we then selected 2,000 geometry maths problems to create our GeoEval-2000 subset. This selection process was guided by the aim to inclusively cover a wide range of basic geometric shapes, ensuring a broad representation of geometry concepts.[1]

4.2.1.2 Backward Data Generation


Forward Question
As shown in the figure, in △ABC, AB = AC, ∠A = 30.0, the perpendicular bisector of AB intersects AC at vertex D, then the degree of ∠CBD is ?

Backward Question
As shown in the figure, in △ABC, AB = AC, ∠A = x, the perpendicular bisector of AB intersects AC at vertex D, then the degree of ∠CBD is ?
The correct answer is 45.0. Now please answer what is the value of x?

Figure 4.1: Example for the backward question.

In contrast to forward problems, backward problems use the answer from forward problems as a starting point, posing a query to determine a specific number that was

[1]Please see Appendix B.1 for the further information of GeoEval-2000 dataset.

part of the forward problems but is concealed in the backward problems [217]. These types of questions are particularly effective in assessing models' capability for multi-step reasoning. Following the methodology of previous research [218], we selected 750 problems from the GeoEval-2000 subset and created corresponding backward questions. This process involved masking a number, the solution of the forward problems, as "X". The prompt "*The correct answer is* ans_{gold}. *Now please answer what is the value of X?*", where ans_{gold} represents the correct answer to the forward problems, is then added. An example of the backward problem is displayed in Figure 4.1.

4.2.1.3 Augmented Data Generation

To evaluate the resilience of current models and mitigate the risk of data leakage that may occur during the pre-training phase, we implement a context learning strategy for rephrasing problems from the GeoEval-2000 subset. Each problems is rephrased into five variant candidates by GPT-3.5 [219], ensuring they retain the original problems's semantic essence while varying in lexical structure. Out of these five alternatives, one is selected randomly to substitute the original problems, forming the GeoEval-aug subset.

4.2.1.4 Hard Data Collection

While the GeoEval-2000 subset comprises geometry problems from a variety of source datasets, it exhibits a lack of diversity in problem categories, notably in solid geometry and analytic geometry. To enhance the diversity of problem categories, we introduce the GeoEval-hard subset, which includes 300 geometry problems specifically focusing on solid geometry and analytic geometry, providing a broader assessment scope. The distinctions between the GeoEval-hard subset and other publicly available datasets are detailed in Table 4.2, demonstrating the unique coverage and complexity of the GeoEval-hard subset in comparison.

The creation of the GeoEval-hard subset begins with web scraping to gather approximately 10,000 authentic geometry problems from online resources. An initial selection is made using a rule-based engine equipped with a keyword list, targeting solid and analytic geometry problems. This step yields around 3,100 potential problems, identified

Dataset	Solid Geometry		Analytic Geometry	
	#solid geometry shapes	#question type	#geometry curve knowledge	#grade
UniGeo [1]	✗	calculate/prove	✗	6-12
GeoQA [61]	✗	calculate	✗	6-12
Geometry3K [213]	✗	calculate	✗	6-12
PGPS9K [5]	✗	calculate/judge	✗	6-12
MathVista(Geometry Part) [12]	✗	calculate/judge	✗	–
MathVista(FunctionQA Part) [12]	✗	calculate/judge	✓	–
GeoEval-hard	✓	judge/calculate/reason	✓	9-12

Table 4.2: Comparison between GeoEval-hard with other public datasets.

as GeoEval-hard-raw. A manual review further narrows these down to 300 problems specifically related to solid and analytic geometry. The cleaning and manual inspection process, is documented in Appendix B.2.

4.2.2 Features of GeoEval

Dataset	*Comprehensive Variety*	*Varied Problems*	*Dual Inputs*	*Diverse Challenges*	*Complexity Ratings*
MathQA [9]	n/a	flat	text	✗	✗
GeometryQA [215]	n/a	flat	text	✗	✗
Geometry3K [213]	n/a	flat	text + diagram	✗	✗
GeoQA+ [61]	n/a	flat	text + diagram	✗	✗
MATH [216]	n/a	flat	text	✗	✗
UniGeo [1]	n/a	flat	text + diagram	✗	✗
PGPS9K [5]	n/a	flat	text + diagram	✗	✗
GeomVerse [214]	n/a	flat	text + diagram	✗	✓
MathVista [12]	4	flat	text + diagram	✗‡	✗
GeoEval	7 + 3 (new)	flat, solid, analytic	text + diagram	✓	✓

Table 4.3: Comparison between GeoEval benchmark and other datasets. Under *Comprehensive Variety*, MathVista and GeoEval stand out as collective datasets, while the rest, denoted as "n/a". GeoEval includes problems from seven public datasets and three newly created ones. *Varied Problems* categorises problems into "flat geometry", "solid geometry", and "analytic geometry", For *Dual Inputs*, "text" signifies problems presented only in text format, whereas "text + diagram" encompasses problems with both texts and diagrams. In *Diverse Challenges*, the symbol ‡ indicates that MathVista introduces three new datasets, which, however, are unrelated to the geometry problem-solving task.

The GeoEval benchmark is specifically designed for assessing the ability of models in resolving geometric maths problems. This benchmark features five characteristics:

Comprehensive Variety, *Varied Problems*, *Dual Inputs*, *Diverse Challenges*, and *Complexity Ratings*, with each attribute exemplified in the Appendix B.3. For an insightful contrast, Table 4.3 offers a comparative analysis of GeoEval against earlier datasets.

- *Comprehensive Variety*: GeoEval consists of a diverse collection of geometry problems sourced from seven most recent datasets. Therefore, the problems in GeoEval cover a wide range of geometric shapes, offering a comprehensive view of varied geometry maths challenges.
- *Varied Problems*: The GeoEval benchmark encompasses three distinct categories of geometry maths problems, namely flat geometry, solid geometry, and analytic geometry.
- *Dual Inputs*: GeoEval features problems in two formats: those accompanied by diagrams and those consisting solely of text. This versatility makes it suitable for evaluating models that process either diagrams or text-based inputs.
- *Diverse Challenges*: In addition to gathering public datasets, GeoEval also generates its own out-of-distribution dataset aimed at addressing data leakage problems. This includes a backward reasoning subset, an augmented subset, and a hard subset, all created by us.
- *Complexity Ratings*: GeoEval is equipped with annotations indicating the complexity level for each problem, serving as a guideline to evaluate models' proficiency in solving these tasks.[2]

4.3 Chapter Summary

In this chapter, we introduce GeoEval, a benchmark specifically developed to assess the geometry problem-solving capabilities of AI models. GeoEval consists of four distinct subsets, each designed to enable a comprehensive evaluation.

[2]Algorithm for classifying complexity is in Appendix B.4.

In the subsequent Chapter 5 and Chapter 6, we evaluate our proposed methods on portions of the GeoEval dataset. Furthermore, Chapter 7 provides an evaluation of the latest LLMs and MMs, assessing their performance on the full GeoEval benchmark.

Chapter 5

GAPS: Geometry-Aware Problem Solver

In Chapter 2.2, we realise that lots of methods were proposed for solving the task of text-based automated numerical reasoning task. However, for the task of solving geometry maths problems, existing approaches often rely on models designed for solving maths word problems, neglecting the unique characteristics of geometry maths problems. Additionally, the current research predominantly focuses on geometry calculation problems, while overlooking other essential aspects like proving.

In this Chapter, we address these limitations by proposing the Geometry-Aware Problem Solver (GAPS) model. GAPS is specifically designed to generate solution programs for geometry maths problems of various types with the help of its unique problem-type classifier. To achieve this, GAPS treats the solution program as a composition of operators and operands, segregating their generation processes. Furthermore, we introduce the geometric elements enhancement method, which enhances the ability of GAPS to recognise geometric elements accurately. By leveraging these improvements, GAPS showcases remarkable performance in resolving geometry maths problems. Our experiments conducted on the UniGeo dataset demonstrate the superiority of GAPS over the state-of-the-art model, Geoformer. Specifically, GAPS achieves an accuracy improvement of more than 5.3% for calculation tasks and an impressive 41.1% for proving tasks. Notably, GAPS achieves an impressive accuracy of 97.5%

on proving problems, representing a significant advancement in solving geometry proving tasks. This chapter is mainly based on my under-review submission on Artificial Intelligence journal (Elsevier).

The subsequent sections of this chapter are structured as follows: Section 5.1 provides an overview of the challenges in solving geometry maths problems and outlines the current limitations in this field. It also details the specific research questions addressed in this chapter and enumerates the contributions of this chapter. In Section 5.2, we briefly introduce how the VL-T5 [4] work, which is used as the text encoder for our proposed GAPs model. Section 5.3 offers an in-depth description of the newly proposed GAPS model. Section 5.4 outlines the experimental setup. Section 5.5 presents the results of our experiments, providing a thorough analysis in the context of the research questions introduced in Section 5.1. The chapter concludes with Section 5.6, which summarises the key findings and insights gained from this chapter.

5.1 Introduction

Deep learning techniques have shown remarkable success in text-based numerical reasoning task using methods such as CoT [98] and MultiHiertt [110]. However, when it comes to automatically solving geometry maths problems, the research is still at an early stage. Recent efforts have attempted to adapt models that succeeded in MWPs to handle geometry maths problems, but only a few methods have yielded satisfactory results. One possible reason for this discrepancy is that, unlike MWPs, geometry maths problems involve additional complexity due to the presence of geometric diagrams. To illustrate this point, consider the typical geometry maths problem shown in Figure 1.1. Effectively solving this problem requires a deep understanding of both the textual information and the accompanying diagrams [61]. Moreover, it is crucial to identify the geometric elements mentioned in the problem text to achieve satisfactory performance [45].

Although researchers have been motivated to tackle geometry maths problems from various angles, the existing approaches can be broadly categorised as utilising pre-trained language models (PLMs) or recurrent neural networks to directly generate solution programs [1,11,61]. Consequently, these methods lack an inherent architectural design that explicitly captures the unique characteristics of geometry maths problems.

Geometry maths problems can be categorised into various types, with two dominant ones being calculation (CAL) problems and proving (PRV) problems (as depicted in Figure 1.1). However, most recent works have primarily focused on CAL problems because other types of geometry maths problems require different domain-specific languages. For example, solution programs for CAL problems involve arithmetic operators and numbers, while solution programs for PRV problems are constructed using geometric theorems and geometric elements. Nevertheless, it is essential to recognise that the mathematical knowledge required to solve CAL problems significantly overlaps with the knowledge needed for PRV problems [1]. This observation underscores the considerable importance of developing a single solver that can generate solution programs for geometry maths problems of all kinds of types. By creating such a versatile solver, we can leverage the shared mathematical knowledge and potentially devise more efficient and effective problem-solving techniques for a wide range of geometry maths problems.

To resolve the difficulties in the automated geometry maths problem solving, this chapter states that using one solver for various geometry maths problems can enhance performance (see Statement (2) in 1.3). Specifically, this chapter introduces the Geometry-Aware Problem Solver (GAPS), a model designed to automatically solve geometry maths problems of different types simultaneously. GAPS leverages the VL-T5 [4] model to obtain unified representations of text and diagrams, which are then used to predict the solution programs.

To achieve a single solution program generator for various problem types, we assume that solution programs hold a unified pattern that consists of operators (arithmetic operators, geometric theorems) and operands (numbers, geometric elements). To handle different problem types, GAPS incorporates a problem-type classifier, which distinguishes between the problem types of various geometry maths problems. By combining the unified solution program pattern with the problem-type classifier, GAPS can independently generate the operators and operands by selecting tokens from different domain-specific languages based on the specific problem types. Leveraging the unified solution program and the problem-type classifier, GAPS achieves the distinct capability of disentangling the process of generating the operators and operands within the solution program. To amplify this framework, we enhance GAPS by incorporating the hierarchical beam search, a decoding technique aligned with the approach of segregating operator and operand generation within GAPS. Furthermore, aiming to augment GAPS' proficiency in identifying geometric elements from the provided problem text, we introduce the geometric elements enhancement method which hinders the special geometry tokens being recognised as unknown tokens.

To validate the statement, this chapter investigates the following research questions (RQ):

- **RQ-5.1 Comparison with Existing Models**: How does the performance of GAPS compare to state-of-the-art models on the UniGeo Cal, UniGeo Prv, PGPS9K, and Geometry3K datasets, specifically in terms of accuracy and comprehensiveness in reasoning?
- **RQ-5.2 Efficiency in Solving Different Types of Geometry Problems**: Given the superior capabilities of GAPS in solving geometry maths problems, how does its proficiency vary across different types of geometry problems?
- **RQ-5.3 Investigation for the model's weakness on certain problem type**: What are the reasons for GAPS achieve better results on solving proving problems than solving the calculation problems? Can the reasons be generalised to evaluate the difficulty level of the geometry maths problems?
- **RQ-5.4 Enhancement through Augmented Training Data**: How does the training of GAPS on merged datasets featuring a variety of domain-specific languages (DSLs) impact its performance and ability to generalise across diverse problem types? And how does the performances are boosted with the help of the problem-type classifier?
- **RQ-5.5 Effectiveness of the problem-type classifier**: How does the integration of a problem-type classifier in GAPS enable the model to handle di-

verse types of geometry maths problems and enhance its adaptability to different domain-specific languages without compromising performance?

- **RQ-5.6 Investigation of the Number of Problem Types**: How does the accuracy in identifying the correct number of problem types by the problem-type classifier influence the benefits and overall performance of the GAPS model?
- **RQ-5.7 Advantages of Unified Solution Program Generator**: How does employing a unified solution program generator, harmonised with a problem-type classifier, allow GAPS to tackle a diverse array of geometry maths problems proficiently, and what are the limitations, if any, of this approach?
- **RQ-5.8 Comparison of the different cache token updating methods**: What is the best approach to update the cache token embeddings? So that it helps the model to fully utilise the results of previous sub-programs?
- **RQ-5.9 Effectiveness of the Hierarchical Beam Search Algorithm**: How does the integration of hierarchical beam search technology enhance GAPS' ability to proficiently select high-quality solution programs compared to other search technologies like greedy search?
- **RQ-5.10 Effectiveness of the geometric element Enhancement Method**: How does the geometric element enhance method that helps the model in selecting the geometric elements as operands for the solution programs? And will the same strategy help the model in selecting numbers from the text?

5.1.1 Contributions

The contributions of this chapter are:

1. We present the GAPS model, which introduces a novel encoder capable of seamlessly processing both text and diagram inputs. Additionally, a geometry-specific program generator is incorporated to create solution programs for geometry maths problems. This integration enables the model to effectively understand geometric diagrams.

2. GAPS introduces a framework to handle a wide range of geometry maths problems without requiring separate program generators for different problem types. This is achieved by inserting a problem-type classifier between the diagram and problem text encoder and the program decoder. This augmentation enhances GAPS' adaptability, allowing it to handle diverse types of geometry maths problems.

3. The geometry-specific program generator divides the generation of operators and operands in the solution program. This innovation presents a challenge for employing beam search decoding strategy. In this paper, we address this challenge by proposing hierarchical beam search as a pivotal enhancement for GAPS. This inventive beam search approach empowers GAPS to produce precise and varied solution programs for geometry maths problems, leading to a performance boost.

4. To comprehensively assess GAPS' capabilities, we conduct extensive experiments on various geometry maths problem datasets, including UniGeo Calculation, UniGeo Proving, PGPS9K, and Geometry3K. The experimental outcomes underscore GAPS' exceptional performance compared to alternative methods designed for solving geometry maths problems.

5.2 Preliminary: VL-T5

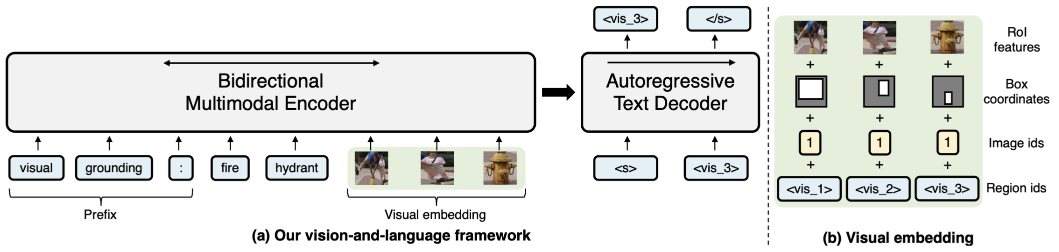


Figure 5.1: An illustration of the VL-T5 model. The image is taken from the original paper [4].

This section outlines the foundational knowledge necessary to grasp the workings of our GAPS model. Central to GAPS is the use of VL-T5, as detailed in [4], which serves as the textual encoder. This differs from the conventional Transformer architecture discussed in Section 2.1.1, particularly in how VL-T5 incorporates additional tokens $\{< vis\_1 >, ..., < vis\_n >\}$ for image representation. Here, the number n denotes the n-th region the image, as depicted in Figure 5.2(b). Each region feature e_i^v is a composite of four distinct types of features: 1. RoI (Region of Interest) object features, 2. RoI bounding box coordinates, 3. image ids (ranging from 1 to 2), and 4. region ids (ranging from 1 to n). The visual embeddings for the image are thus considered as a cumulative representation of all these regional features e_i^v.

In line with other transformers, VL-T5 processes the combined textual and visual embeddings as inputs to the transformer block to generate joint representations. For more detailed information on the functioning of each transformer block, please refer to Section 2.1.1.3.

5.3 Approach

This section describes the details of the Geometry-Aware Problem Solver (GAPS), which is tailored specifically for solving geometry maths problems. The architecture of GAPS is illustrated in Figure 5.2. GAPS employs a joint diagram and problem text encoder to create unified representations for both textual and diagram modalities. These unified text-diagram representations are then utilised by the geometry-specific program generator to produce the solution program, with a separation of the operators and operands. To enhance the accuracy of the generated solution programs, we introduce a hierarchical beam search for the geometry-specific program generator. Moreover, to handle various types of geometry maths problems, we incorporate a problem-type classifier within GAPS, which enables the use of a single geometry-specific program generator to produce solution programs for different problem types. The generator employs masks according to the problem-type classifier to select the appropriate symbols from the domain-specific language defined for each type of geometry maths problem.

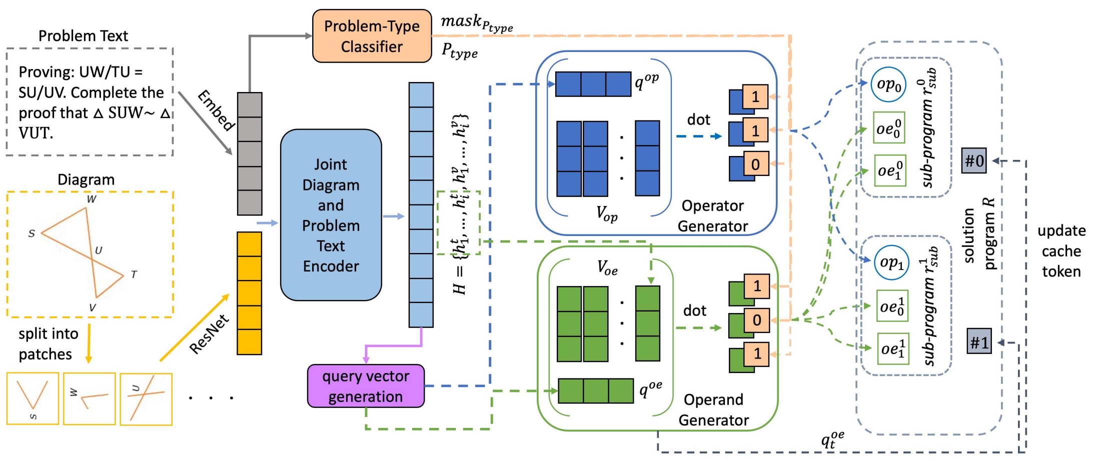


Figure 5.2: First, the diagram is divided into patches, and along with the problem text, both are transformed into vectors. These vectors are then concatenated and passed to the joint diagram and problem encoder, producing the joint representations denoted as H. Subsequently, the geometry-specific program generator starts to generate the solution program, each solution program is represented by several sub-program r^i_{sub}. Each sub-program r^i_{sub} contains one operator op_i and corresponding operands $\{oe^i\}$, where the query vector q^{op} is used for generating the operator op^i, and the query vector q^{oe} is used for generating operands in the sub-program. After each sub-program is generated, the corresponding cache token is updated by replacing its vector by the q^{oe}_t, which is the query vector used for generating the last operand in the sub-program. Specifically, P_{type} is used to produce the mask to multiply with the probabilities of operators and operands, thus influencing GAPS to select op and oe from the domain-specific language (DSL) of the corresponding problem type. This mechanism enables GAPS to adapt and solve different kinds of geometry maths problems effectively.

5.3.1 Task Definition

The objective of our task is to generate the correct solution program $\mathcal{R}$ for a given geometry maths problem $\mathcal{P}$ accompanied by a geometric diagram $\mathcal{D}$. Each problem text $\mathcal{P}$ belongs to a specific problem type P_{type}. The solution program $\mathcal{R}$ consists of both operators op and operands oe. The operator op comprises various arithmetic operations (e.g., $+, -$) as well as advanced functions such as trigonometric functions and formulas for calculating areas. On the other hand, the operands oe can be pre-defined constant numbers or numbers (num) and geometric elements (ele) extracted from the problem text $\mathcal{P}$. Please refer to Table 5.1 for a detailed explanation of the notation symbols used in the context.

Notation	**Description**
$\mathcal{P}$	geometry maths problem text
$\mathcal{D}$	geometric diagram
$\mathcal{R}$	solution program
P_{type}	problem type
r_{sub}^{t}	the t-th sub-program of $\mathcal{R}$
op_t	the operator in r_{sub}^{t}
oe_i^t	the i-th operand in r_{sub}^{t}
DSL$_{op}$	domain-specific language defined for operator
DSL$_{oe}$	domain-specific language defined for operand
$\#t$	cache token refers to r_{sub}^{t}
num	numbers from $\mathcal{P}$
ele	geometric elements from $\mathcal{P}$

Table 5.1: The notations used for the task definition.

5.3.1.1 Normalise the Representation of the Solution Program

We represent the solution program $\mathcal{R}$ as a combination of operators and operands, adhering to the output format of GAPS's geometry-specific program generator. Moreover, the solution program $\mathcal{R}$ is divided into a set of sub-programs r_{sub}, each containing one operator and its corresponding operands. Formally, a solution program $\mathcal{R}$ is defined as $\mathcal{R} := \{r_{\text{sub}}^{t}\}_{t=0}^{T}$, where T represents the total number of sub-programs in the solution program $\mathcal{R}$, and r_{sub}^{t} refers to the t-th sub-program within the solution program. Furthermore, each sub-program r_{sub}^{t} is defined as $r_{\text{sub}}^{t} := (op_t, \{oe_i^t\}_{i=0}^{I})$, where op_t denotes the t-th operator in the solution program $\mathcal{R}$, and it is the sole operator included in the sub-program r_{sub}^{t}. The set of $\{oe_i^t\}_{i=0}^{I}$ is associated with op_t, comprising a total count of I. For a concrete example, please refer to Figure 5.3.

5.3.1.2 Domain Specific Language

As stated earlier, the geometry maths problem text $\mathcal{P}$ lack explicit domain-specific knowledge regarding operators and operands, making it necessary to define a domain-specific language (DSL) to aid the GAPS model in decoding. In particular, the DSL

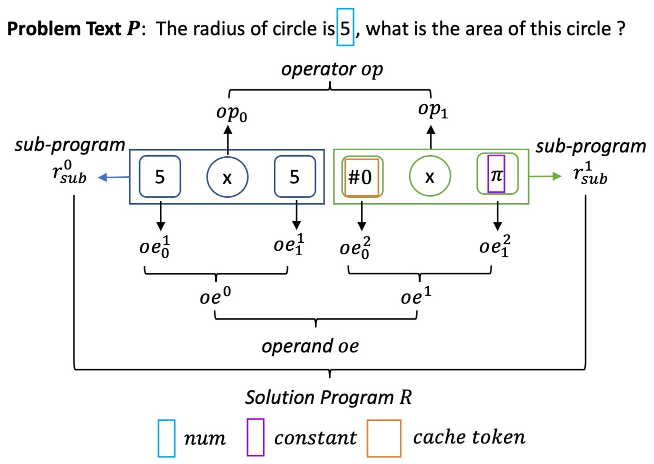


Figure 5.3: An example of normalised representation of the solution program.

for operators (DSL$_{op}$) encompasses basic arithmetic operations, geometry theorems (e.g., Corresponding Angles Theorem), and geometry calculations (e.g., Circle_R_Area). Additionally, the DSL for operands (DSL$_{oe}$) includes constant numbers (e.g., the value of π) and cache token $\#i$ that represents the result of the previous sub-program. For a comprehensive list of the DSL, please refer to the original dataset or the code link provided by us.

5.3.2 Joint Diagram and Problem Text Encoder

We opt for VL-T5 [4] as our encoder due to its capability to uniformly embed both text and diagram modalities without requiring any modifications to the transformer architecture. Specifically, the geometry problem text $\mathcal{P}$ is tokenized into tokens $\{t_1, ..., t_i\}$ and mapped into learned text embeddings $\{e^t_1, ..., e^t_i\}$, where i is the number of the tokens in $\mathcal{P}$. Additionally, the diagram $\mathcal{D}$ is divided into n patches, each of which is embedded into visual embeddings $\{e^v_1, ..., e^v_n\}$. To create the joint representations, the text embeddings $\{e^t_1, ..., e^t_i\}$ and visual embeddings $\{e^v_1, ..., e^v_n\}$ are concatenated and fed into the VL-T5 transformer. The output of this process is the contextualised joint representations $H = \{h^t_1, ..., h^t_i, h^v_1, ...h^v_n\}$, where $H \in \mathbb{R}^{(i+n)\times h}$, and h represents the dimension of the embeddings.

5.3.2.1 Ggeometric Elements Enhancement

The tokenizer employed by VL-T5 often converts geometric elements (e.g., $\triangle$) in the problem texts to unknown tokens, making it challenging for the program generator to differentiate between elements that share the same points (e.g., $\angle SUW$ and $\triangle SUW$). To address this issue, we propose replacing geometric elements (e.g., $\triangle$) with geometry terminology tokens (e.g., "triangle"). Moreover, we introduce a straightforward yet useful technique called "geometric elements enhancement" to enhance the accuracy of selecting geometric elements ele from the problem text as operands. Specifically, we repeat and append these geometry terminology tokens to the problem text $\mathcal{P}$. By doing so, the model becomes capable of distinguishing such elements accurately. In our ablation study, we demonstrate the effectiveness of this enhancement technology in improving the model's performance.

5.3.3 Problem-Type Classifier

In order to handle different types of geometry maths problems with varying DSLs, it becomes necessary to combine the distinct DSLs into a unified one. However, this amalgamation unavoidably increases the complexity of solving geometry maths problems, as it expands the search space of the decoding vocabulary. To address this challenge and alleviate the difficulty of generating the solution programs, we draw inspiration from the observation that neural models find it easier to perform discriminative tasks compared to generative tasks. As a solution, we introduce the problem-type classifier, which serves to distinguish between problem types for different geometry maths problems. The problem-type classifier takes the contextualised representations $H = \{h_1^t, ..., h_i^t\}$ of the problem text as input and subsequently generates probability distributions for all problem types:

$$\mathbb{P}(P_{\text{type}}|h_1^t, ..., h_i^t) = \text{softmax}(W_1 \cdot \text{sum}(h_1^t, ..., h_i^t, \text{dim} = 1)) \tag{5.1}$$

where $W_1 \in \mathbb{R}^{c \times h}$ is trainable parameters, an c is the number of problem types. Following the classification result P_{type}, the problem-type classifier generates a mask that converts the probabilities of operators and operands not belonging to P_{type} into zeros. This masking mechanism ensures that irrelevant symbols are not generated, thus preventing the introduction of erroneous symbols in the solution program. By leveraging the problem-type classifier, GAPS can effectively address geometry maths problems of various problem types with a single program generator. This approach simplifies the training process as there is no need to separately train multiple program generators for different problem types, streamlining the model and promoting its versatility across different geometry maths problems.

5.3.4 Geometry-Specific Program Generator

With the utilisation of the joint diagram and problem text encoder, coupled with the problem-type classifier, the geometry-specific program generator takes the contextualised combined representations denoted as H as its input. It proceeds to construct the solution program $\mathcal{R}$ by iteratively generating its sub-program r_{sub} over multiple steps, where each step is represented as i. Building upon the insights from prior research [220], which highlighted the advantages of segregating the generation process for operators and operands, our geometry-specific program generator adopts an alternating approach. Specifically, during the generation of each sub-program r^i_{sub} at the i-th step, the generator switches between generating the operators and the operands within the sub-program. This strategy empowers the GAPS model to proficiently devise accurate solution programs. It capitalises on the contextual cues from both the problem text and diagram, while also drawing upon the problem-type classifier to ensure the appropriateness of the generated symbols with respect to the specific problem type.

5.3.4.1 Generating Sub-Program Step-by-Step

Given the inherent variability in decoding symbols s for numbers and geometric elements across different input geometry problems, it's important to note that each index within the decoding vocabulary doesn't consistently correspond to the same symbol s. Moreover, the number of decoding options for both numbers and geometric elements within each input is contingent on the specific content of the original problem text.

Consequently, the conventional approach of devising a fixed-length decoding vocabulary, commonly employed in language models, is unfeasible in this context.

In drawing inspiration from the principles of pointer networks [145] and cross-attention architectures [2], we introduce the concept of "decoding values vectors". This concept facilitates the program generator in selecting the appropriate symbol s. Concretely, for a given input geometry problem, the decoding values vectors encompass the representations of all decoding symbols. These vectors are formulated as $V = W_v(e_s)$, with e_s representing the embedding vectors for each symbol. Notably, e_s encompasses embeddings for both operators and operands defined within the domain-specific language (DSL), and these embeddings are amenable to training. Additionally, e_s incorporates vectors corresponding to numbers or geometric elements present in the problem text of the input geometry problem. These are formulated as h_i^t, with i denoting the positional index of a number or geometric element within the geometry maths problem text $\mathcal{P}$.

In order to accurately choose the appropriate symbol s as the output for the current step, we formulate a query vector $q_{s_{t-1}}$ that encapsulates the information derived from both the previously generated symbol s_{t-1} and the combined representation H:

$$q_t^{op,oe}, h_t = \mathrm{GRU}_{op,oe}(\mathrm{ReLU}(W_2[P_{aware} : v_{s_{t-1}}]), h_{t-1}) \tag{5.2}$$

where $W_2 \in \mathbb{R}^{h \times 2h}$ is trainable parameters, and ReLU is the activation function, and h_{t-1} is the hidden state of the GRU from the previous step (by default, h_0 is initialised as zero vector). Since we separate the generation of operators and operands, two GRUs with different parameters are adopted. The $v_{s_{t-1}}$ refers to the value vector of the previous generated symbol s_{t-1}. Notably, P_{aware} encodes the weighed sum information of H, which is calculated by:

$$P_{aware} = \sum_i a_i h_i^{t,v} \tag{5.3}$$

where $h_i^{t,v}$ is the representation vector of the i-th token in H, and a_i is the weight, which is calculated through:

$$a_i = \frac{\exp(\text{score}(v_{s_{t-1}}, h_i^{t,v}))}{\sum_L \exp(\text{score}(v_{s_{t-1}}, h_j^{t,v}))} \tag{5.4}$$

$$\text{score}(v_{s_{t-1}}, h_i^{t,v}) = (v_{s_{t-1}}^T W_3) \cdot (W_4 h_i^{t,v}) \tag{5.5}$$

where $W_3 \in \mathbb{R}^{h \times h}$ and $W_4 \in \mathbb{R}^{h \times h}$ are trainable parameters, and L is the total number of problem text tokens and diagram patches.

Subsequently, for the generation of the t-th symbol s_t within the sub-program r_{sub}^i, we employ the query vector $q_t^{op,oe}$ to perform a dot product with the decoding values vectors V. This operation yields a probability distribution across all potential decoding candidates:

$$\mathbb{P} = \text{softmax}(q_t^{op,oe} \cdot (W_5 \cdot V_{op,oe})) \times \text{mask}_{P_{type}} \tag{5.6}$$

where $W_5 \in \mathbb{R}^{h \times h}$ is trainable parameters. The symbol with the highest probability among $\mathbb{P}$ is selected as the predicted operator for the current step. When generating the initial symbol s_0, since each sub-program comprises a single operator alongside multiple operands, the decoding value vector is limited to V_{op}. This vector exclusively contains operators as defined within the DSL, facilitating the generation of op_i.

Subsequent to generating the operator op_i for the sub-program r_{sub}^i, GAPS replaces the decoding values vectors to V_{oe}, which exclusively includes the constants specified within the DSL, as well as numbers (or geometric elements) present within the problem text. This adjustment allows GAPS to proceed with the completion of the associated operands $\{oe^i\}$ within the sub-program r_{sub}^i. The generation of operands concludes when the "end of sequence" token (denoted as "*eos*") is produced.

The sub-program r_{sub}^i is considered finished when both its operator op^i and operands $\{oe^i\}$ have been generated. Following this, the program generator iterates through the same process outlined above to generate the following sub-programs $r_{sub}^{t>i}$.

5.3.4.2 Using one Program Generator for Different Problem Types

In order to streamline the process and avoid the need for distinct program generators for varying problem types in geometry problems, we incorporate all symbols from the DSL into the decoding values vectors V. To ensure that the chosen symbol adheres to the $\text{DSL}_{P_{type}}$ specific to the given problem type P_{type}, we utilise the mask $\text{mask}_{P_{type}}$, generated by the problem-type classifier. This mask effectively eliminates probabilities associated with irrelevant symbols, ensuring that only valid symbols are taken into account during the generation process (as illustrated in Equation 5.6). During training, the correct P_{type} information is provided by the ground truth. During the inference stage, the problem-type classifier (as seen in Equation 5.1) is deployed to determine the P_{type} for the geometry maths problem. This strategic approach empowers the GAPS model to employ a singular program generator capable of addressing diverse geometry maths problems, thereby enhancing the efficiency of the overall architecture.

5.3.4.3 Updating the Cache Token

During the generation of sub-program r^i_{sub}, its operands $\{oe^i\}$ have the potential to be derived from the executable result obtained from the preceding sub-program $r^{<i}_{sub}$. To enable the operand generator to effectively leverage results from earlier sub-programs, GAPS employs the cache token $\#i$ within the DSL_{oe}, drawing inspiration from previous work [220]. This cache token symbolises the executable outcome of the i-th sub-program r^i_{sub}. Diverging from other operands, the content encapsulated by $\#i$ varies based on the specific sub-program it references. Consequently, GAPS needs to update its embeddings once the generation of sub-program r^i_{sub} is concluded.

To this end, we explore several approaches for updating the cache token $\#i$:

- Employing the q^{oe}_t as its embedding: q^{oe}_t signifies the query vector utilised in generating the last operand for sub-program r^i_{sub}.

- Utilising q^{op}_0 as its embedding: q^{op}_0 represents the query vector applied to generate the operator for sub-program r^i_{sub}.

- Incorporating e_{op} as its embedding: e_{op} corresponds to the embedding of the operator generated for sub-program r^i_{sub}.

By using the cache token $\#i$, GAPS enhances its ability to make contextually informed decisions during operand generation, thereby improving overall performance.

5.3.4.4 Hierarchical Beam Search for Operators and Operands

To ensure the production of high-quality solution programs and mitigate the possibility of sub-optimal solutions, GAPS enhances its geometry-specific program generator with the hierarchical beam search. This beam search approach aligns well with the design of segregating generations between operators and operands. Here is an outline of how the hierarchical beam search operates (refer to algorithm in Appendix C.1):

Let "*ops*" represent the set of all possible operators, and "*bs*" represent the beam size. Let $\mathcal{R}$ represent the final solution program.

- **Operator Generation:** At each step of generating the sub-program, the hierarchical beam search selects the number of *bs* operators from *ops* as candidates. Let $op_{candidates}$ be the set of operator candidates and output probabilities.

- **Operand Generation:** For each operator candidate $op \in op_{candidates}$, the operand generator takes *op* as input and produces the number of *bs* sequences of operands. Let $oe_{candidates}$ represent the set of operand sequences generated for operator candidate *op*.

- **Score Computation:** Once completing the generation of operands for each operator candidate, the output probabilities between operators and their corresponding operands are summed up to obtain scores for each combination. Let $score_{op\_oe}$ denote the scores of all operator candidates with their operand sequences.

- **Candidate Selection:** The $score_{op\_oe}$ are sorted in descending order. The algorithm selects the top *bs* operator candidates along with their corresponding operand sequences.

- **Iteration:** Repeat the above steps until the generation of the solution program $\mathcal{R}$ is complete. In each iteration, new operator candidates and operand sequences are selected based on the scores computed in the previous iteration.

By using the hierarchical beam search, GAPS can effectively explore different combinations of operators and operands, leading to the selection of high-quality solution programs. This approach is instrumental in improving the overall performance and accuracy of the model in solving geometry maths problems.

5.3.5 Training Objective

During the training phase, the GAPS model's parameters are updated by minimising the combined negative log-likelihoods. This sum encompasses the negative log-likelihoods associated with several components, including the ground truth geometry maths problem type P_{type}, the ground truth operator at each step in the ground truth solution program, and the ground truth operand for each sub-program in the ground truth solution program.

$$L = -\{\log\mathbb{P}(P_{\text{type}}|H) + \frac{1}{L_{prog}}\sum_{i=0}^{L_{prog}}\log\mathbb{P}(op_i) + \frac{1}{L_{prog}}\sum_{i=0}^{L_{prog}}\frac{1}{L_{sub}}\sum_{j=0}^{L_{sub}}\log\mathbb{P}(oe_j^i)\} \quad (5.7)$$

where L_{prog} and L_{sub} refer to the length of the golden solution program and the length of the golden sub-program. op_i is the i-th golden operator in the solution program, and oe_j^i is the j-th golden operand in the i-th sub-program of the solution program.

5.4 Experimental Setup

This section is dedicated to detailing the datasets selected for the evaluation of our methods. It also enumerates the baseline models and metrics employed in our experimental analysis. Lastly, an in-depth explanation of the implementation specifics of our methods is provided.

5.4.1 Datasets

We conducted an extensive evaluation of our model using the UniGeo dataset [1], the latest dataset specifically designed for assessing mathematical reasoning ability in geometry maths problems. The UniGeo dataset is an extension of the GeoQA dataset [11], encompassing 4,998 CAL problems from GeoQA and an additional 9,543 new PRV problems. For each problem in the UniGeo dataset, annotated step-by-step solution programs are provided. The UniGeo dataset further categorises each problem into one of eight sub-tasks, along with the corresponding number of problems in each sub-task (shown in parenthesis): Angle (2,737), Length (1,869), Other (392), Parallel (443), Triangle (3,035), Quadrangle (1,704), Congruent (2,808), and Similarity (1,553).

To maintain consistency with the authors of the UniGeo dataset, we followed their approach to split the dataset into training, validation, and test subsets, with respective ratios of 0.7, 0.15, and 0.15. This dataset serves as a comprehensive benchmark for evaluating our model's performance in tackling diverse geometry problem types and assessing its mathematical reasoning capabilities.

5.4.2 Baselines

- **No Solution Programs** we selected three methods known for their effectiveness in multi-modal reasoning tasks: FiLM [195], RN [221], and MCAN [222]. It is important to note that these baselines differ from our proposed approach in that they do not generate solution programs to solve geometry maths problems. Instead, they treat the problem as a classification task.

- **Encode Text and Diagram Separately** we also compare our model with Seq2Prog [9] and BERT2Prog [78]. Seq2Prog employs a GRU-based as text encoder, while BERT2Prog utilises BERT as the text encoder. For the diagram encoder, they adopt the ResNet [6]. Both Seq2Prog and BERT2Prog follow a similar approach of directly concatenating text embeddings and diagram embeddings as input to an LSTM decoder.

- **Geometry-Specific Models** We also compare the two previous models specified for solving geometric problems: NGS [11] and Geoformer [1]. NGS adopts an LSTM [133] for encoding text and ResNet-101 [6] for diagram encoding. The model combines information from both modalities through a co-attention reasoning mechanism. On the other hand, Geoformer is developed based on VL-T5 [4] and is pre-trained on maths expressions to enhance its performance.

5.4.3 Evaluation Metrics

To maintain consistency with previous works [1,11], we employ top-10 accuracy as the evaluation metric for both CAL problems and PRV problems. Top-10 accuracy measures the percentage of the top 10 predicted solution programs that contain the ground truth program. By using these evaluation metrics, we can gauge how well our model performs in comparison to the previous state-of-the-art models, for solving both CAL problems and PRV problems. This comparison enables us to assess the model's proficiency in generating accurate solution programs for various geometry maths problems.

5.4.4 Implementation Details

The experiment code was written in PyTorch [203] and Huggingface [204]. We opted for the pre-trained T5-base embedding [87] to represent the tokens in the problem text, and we utilised ResNet-101 [6] as the encoder for diagrams. In addition, the same diagram processing method as previous work [11] was used, which changes the background colour to white and resizes each diagram to 224×224. The model was trained on a single NVIDIA A100 GPU (40G VRAM) for 100 epochs, with a learning rate of 2e-4 and a batch size of 40. The model's weights were optimised by Adam optimiser [205]. We used teacher-forcing during training, where the probabilities using the ground-truth symbols for different epochs are: 0.0 when the epoch is less than 10, 0.1 when the epoch is less than 20, 0.5 when the epoch is less than 30, 0.8 when the epoch is less than 40, 0.9 when the epoch is less than 100.

5.5 Experimental Results and Discussions

5.5.1 Comparison with the State-of-the-art Methods (RQ-5.1)

		CAL (%)			PRV (%)					
Methods	**Data**	All	Angle	Length	All	Par.	Tri.	Qua.	Con.	Sim.
FiLM◇	CAL	31.7	34.0	29.7	-	-	-	-	-	-
RN◇	CAL	38.0	42.8	32.5	-	-	-	-	-	-
MCAN◇	CAL	39.7	45.0	34.6	-	-	-	-	-	-
Seq2Prog†	CAL	54.2	66.4	38.5	-	-	-	-	-	-
BERT2Prog†	CAL	54.7	65.8	42.1	-	-	-	-	-	-
NGS‡	CAL	56.9	71.5	49.1	-	-	-	-	-	-
Geoformer‡	CAL	60.3	71.5	49.1	-	-	-	-	-	-
GAPS (Ours)	CAL	65.4 *(+5.1)*	77.7 *(+6.2)*	50.4 *(+1.3)*	-	-	-	-	-	-
BERT2Prog†	PRV	-	-	-	48.0	15.5	48.1	28.5	49.5	77.6
NGS‡	PRV	-	-	-	53.2	13.2	56.6	29.8	57.1	79.4
Geoformer‡	PRV	-	-	-	55.7	13.2	56.6	29.8	57.1	79.4
GAPS (Ours)	PRV	-	-	-	**97.8** ***(+42.1)***	**86.1** ***(+72.9)***	100.0 *(+43.4)*	99.6 *(+69.8)*	**99.5** ***(42.4)***	**92.0** ***(+12.6)***
BERT2Prog†	UniGeo	52.0	63.1	39.2	48.1	15.4	48.0	31.7	49.5	75.1
NGS‡	UniGeo	51.9	63.6	38.8	47.4	11.2	46.9	31.3	48.3	77.6
Geoformer‡	UniGeo	62.5	75.5	48.8	56.4	19.4	69.4	20.4	60.3	75.0
GAPS (Ours)	UniGeo	**67.8** ***(+5.3)***	**78.2** ***(+2.7)***	**54.6** ***(+5.8)***	97.5 *(+41.1)*	85.5 *(+66.1)*	**100.0** ***(+30.6)***	**99.6** ***(+79.2)***	99.1 *(+38.8)*	90.7 *(+15.7)*

Table 5.2: The top-10 accuracy (%) comparison between baselines and our proposed GAPS model. ◇, †, and ‡ refer to "No Solution Programs", "Encode Text and Diagram Separately", and "geometry-specific Models", respectively. We report performances separately when training models on only CAL problems, only PRV problems, and the entire UniGeo problems (see column header "Data"). Numbers in bold refer to the highest accuracy in the specific data subset. In addition, we report the accuracy differences between GAPS and the previous state-of-the-art model (Geoformer) in the parenthesis. Except for the results of Seq2Prog are taken from [11], other baselines' results are taken from [1].

5.5.1.1 Results when trained on both UniGeo CAL and PRV

We present the comprehensive results of our proposed GAPS model and the baseline models in Table 5.2. Notably, GAPS outperforms all models trained on the entire UniGeo dataset, achieving the highest accuracy scores. Particularly, GAPS demonstrates a remarkable increase of 5.3% in accuracy on CAL problems and a substantial improvement of 41.1% on PRV problems compared to Geoformer, the previous state-of-the-art method. This highlights GAPS' superiority in effectively solving geometry maths problems. Furthermore, it is worth noting that both GAPS and Geoformer utilise VL-T5 as encoders. However, GAPS' programs generator exhibits superior performance in producing solution programs for geometry maths problems, indicating its suitability for this specific task. Additionally, GAPS outperforms NGS by 15.9% on CAL problems and an impressive 50.1% on PRV problems. This showcases the significance of designing complex feature fusion techniques to enhance reasoning ability in geometric problem-solving tasks. Finally, the unsatisfactory performance of BERT2Prog emphasises the essential role of step-by-step solution programs, reinforcing their indispensability in tackling geometry maths problems effectively. Overall, the results affirm GAPS' outstanding performance and effectiveness in generating accurate and high-quality solution programs for geometry maths problems, outperforming other state-of-the-art models in this domain.[1]

5.5.1.2 Results when solely trained on UniGeo CAL

When evaluating models trained solely on CAL problems, GAPS continues to exhibit superiority over the selected baselines. Compared to FiLM, RN, and MCAN, GAPS achieves an accuracy approximately twice as high as theirs. This underscores the importance of designing a solution program generation approach tailored for solving geometry maths problems specifically. Furthermore, GAPS outperforms Geoformer on all sub-tasks of CAL problems, a trend consistent with their performance when trained on the entire UniGeo dataset. Moreover, models that uniformly encode text and diagram information, such as GAPS and Geoformer, consistently achieve better results compared to models that naively concatenate embeddings (Seq2Prog, BERT2Prog) or use co-attention mechanisms (NGS). This suggests that performance improvements are observed when using complex feature fusion techniques to align text and diagram information effectively. The findings emphasise the superior capabilities of GAPS in solving geometry maths problems and the advantages of employing sophisticated fusion methods for achieving accurate and comprehensive reasoning in geometry maths tasks.

[1]Please see Appendix 5.5.3.1 for the comparison between top-1 and top-10 accuracy (%) on CAL problems and PRV problems achieved by Geoformer and GAPS.

5.5.1.3 Results when solely trained on UniGeo PRV

When evaluating models trained solely on PRV problems, GAPS consistently outperforms the baselines with competitive advantages. Particularly, GAPS achieves an impressive 42.1% higher accuracy than the previous state-of-the-art method, Geoformer. Compared to GAPS trained on the entire UniGeo dataset, GAPS trained on PRV problems performs slightly better, with a 1.0% accuracy increment. Remarkably, GAPS achieves an overall performance of nearly 100% accuracy in PRV problems, highlighting its exceptional proficiency in solving this problem type. The notable difference of 29.7% between GAPS' performance on CAL problems and PRV problems indicates that GAPS excels in solving the latter type of problem. In the upcoming section, we provide an in-depth analysis to understand the reasons behind the disparity in GAPS' performance across these two problem types.

5.5.2 Comparison of performances according to the sub-program types (RQ-5.2)

Figure 5.4 presents a radar chart illustrating the performance of our proposed GAPS model and the baseline models in CAL problems, PRV problems, and their respective sub-tasks. The radar chart showcases the top-10 accuracy scores for each model in different problem types and sub-tasks. From the radar chart, it is evident that GAPS outperforms the baselines in both problem types and their corresponding sub-tasks in terms of top-10 accuracy. Notably, GAPS demonstrates advantages, particularly in the PRV problems.

5.5.3 Investigation for the Model's Weakness on Certain Problem Type (RQ-5.3)

5.5.3.1 Performance Comparison with Geoformer Using Top-1 and Top-10 Accuracy Metrics

Table 5.3 presents a comparison between our GAPS model and the previous best approach, Geoformer, using a stricter metric, top-1 accuracy, in addition to the top-10

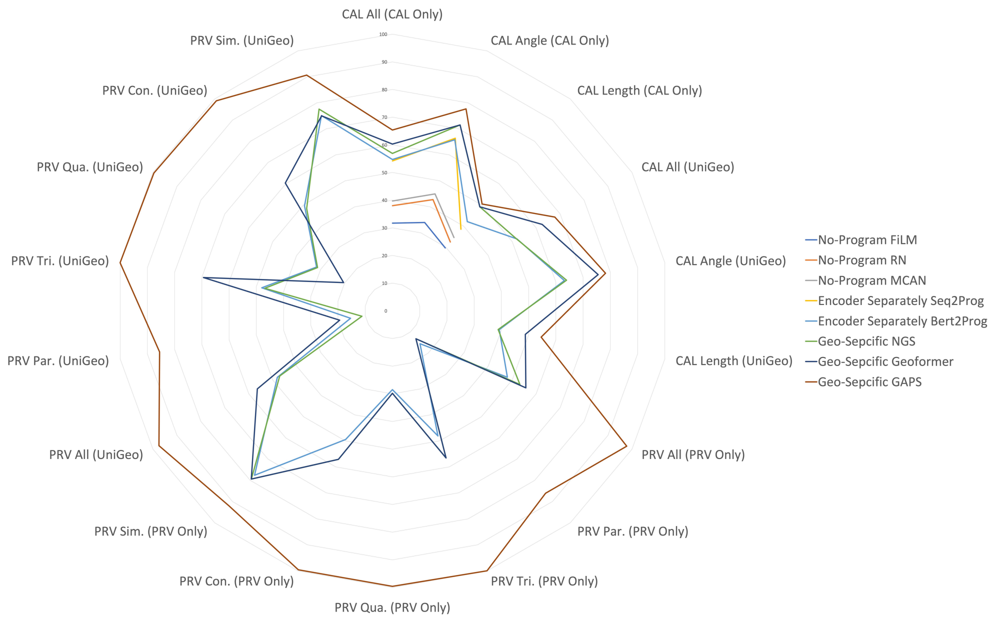


Figure 5.4: Overall results comparison in top-10 accuracy between GAPS and baselines on all sub-tasks by the radar chart.

	CAL (%)		PRV (%)	
Methods	top1	top10	top1	top10
Geoformer	46.8	62.5	51.3	56.4
GAPS (Ours)	42.1	67.8	97.5	97.5

Table 5.3: Comparison between top-1 and top-10 accuracy (%) on CAL problems and PRV problems achieved by Geoformer and GAPS, when training models on the entire UniGeo dataset.

accuracy. The table shows the top-1 and top-10 accuracy scores achieved by GAPS and Geoformer on CAL problems and PRV problems. Consistent with the results using top-10 accuracy, GAPS maintains its superiority over Geoformer on PRV problems when evaluated using top-1 accuracy. Specifically, GAPS achieves a remarkable 46.2% improvement in top-1 accuracy compared to Geoformer, further highlighting its proficiency in solving PRV problems. However, when evaluating CAL problems using top-1 accuracy, our proposed GAPS model exhibits a decrease in performance of 4.7% compared to Geoformer. This contrasts with the results obtained using top-10 accuracy, where GAPS outperforms Geoformer. This discrepancy suggests that GAPS may not always achieve the top predicted solution program in CAL problems, although it still excels in identifying the correct solution program among the top 10 predictions. The variation in performance on CAL problems under different evaluation metrics warrants further investigation in the subsequent analysis to gain a deeper understanding of GAPS' performance characteristics across different problem types.

5.5.3.2 Analysing Prediction Consistency: Top-1 and Top-10 Accuracy Comparison

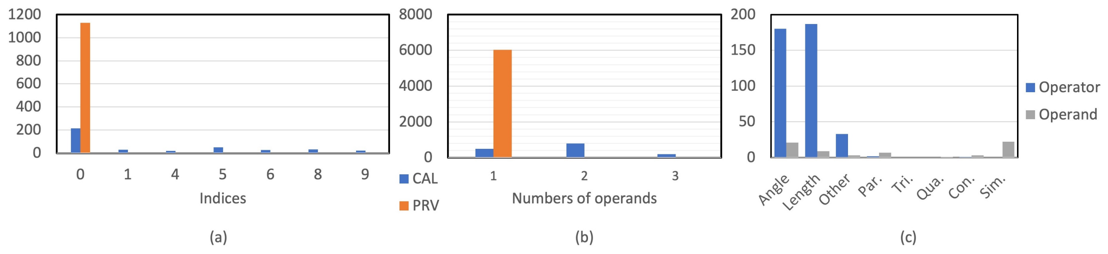


Figure 5.5: (a) Numbers of occurring indices of correct predictions in top-10 predictions candidates. We discard 2th, 3th, and 7th indices because numbers are zero. (b) Distributions of number of operands following each operator in ground truth solution programs from CAL problems and PRV problems. (c) The number of incorrect predictions due to wrong operators or wrong operands generated by GAPS in all sub-tasks. The "Angle", "Length", and "Other" belong to CAL problems, and the others belong to PRV problems.

Table 5.3 reveals a significant discrepancy in the performance of GAPS on PRV problems compared to CAL problems, in terms of top-1 and top-10 accuracy. Interestingly, the top-1 and top-10 accuracy achieved by GAPS on PRV problems are identical, indicating that the model consistently makes correct predictions within the top 10 candidates for these types of problems. In contrast, on CAL problems, there is a considerable difference of 25.7% between the top-1 and top-10 accuracy scores. To investigate the reasons behind this inconsistency, we conduct a detailed analysis by counting the occurrence of correct predictions among the top 10 candidates. Figure 5.5(a) visually presents the distribution of correct prediction indices for both PRV problems and CAL problems. From Figure 5.5(a), we make a significant observation: for PRV problems, all indices of correct predictions are clustered around the 0th index, indicating that the correct solution program is consistently ranked first among the top 10 predictions by GAPS. This finding provides an explanation for the identical top-1 and top-10 accuracy scores obtained by GAPS on PRV problems. In contrast, for CAL problems, the indices of correct predictions are dispersed across several positions within the top 10 predictions. This scattering of correct predictions leads to the difference between the top-1 and top-10 accuracy scores for CAL problems.

5.5.3.3 Analysing Performance Discrepancy: Complexity of CAL and PRV Problems

Table 5.2 presents the top-10 accuracy comparison between the baselines and our proposed GAPS model. An interesting observation is the substantial performance difference in GAPS between CAL problems and PRV problems. This discrepancy could be attributed to the inherent complexity of CAL problems, which typically involve more intricate reasoning processes. To investigate this further, we refer to previous research [220] that highlights the significance of the number of operands following each operator in determining the complexity of solution programs. As such, we plot the distribution of the number of operands following each operator in the ground truth programs for both CAL and PRV problems Figure 5.5(b). From Figure 5.5(b), it is evident that all operators in PRV problems have only one following operand. Conversely, CAL problems exhibit greater variance in the number of operands, ranging from 1 to 3, per operator. This observation confirms our hypothesis that the complexity of solution programs is a key factor influencing GAPS' differing performances on the two problem types. PRV problems, with simpler solution program structures, are more effectively handled by GAPS, resulting in higher accuracy. However, the greater complexity of CAL problems poses a greater challenge, leading to a comparatively lower accuracy for this problem type.

Additionally, we posit that generating correct operators poses a more challenging task than generating correct operands. The complexity of operator generation arises from the necessity to comprehend the semantic meaning of the text and diagrams while also manipulating relevant theorem knowledge. On the other hand, generating operands primarily requires matching entities and numbers in the text and employing finite domain knowledge, such as constants. In line with this hypothesis, we analyse and present the number of wrong predictions caused by mistaken operators or operands in Figure 5.5(c). The findings reveal that in CAL problems, the majority of incorrect predictions are attributed to wrong operators. Conversely, in PRV problems, the incorrect predictions are more frequently due to wrong operands. This disparity in incorrect predictions further validates our hypothesis, demonstrating that generating solution programs for CAL problems is indeed more intricate than generating those for PRV problems. The complexity associated with CAL problems poses challenges for the model, leading GAPS to be more prone to fail in generating correct solution programs for this problem type.

Indeed, our analysis of the difficulty level between CAL problems and PRV problems clearly indicates that CAL problems pose a greater level of diversity and challenge in their solution programs. This inherent complexity in CAL problems explains the better performance achieved by GAPS on the PRV problems than on the CAL problems.

5.5.4 Dataset Augmentation with PGPS9K: A Comparative Analysis (RQ-5.4)

Due to the presence of the unified solution program pattern and the problem-type classifier, GAPS can effectively address a wide range of geometry maths problems using just one solution program generator, all while maintaining the model's performance.

Furthermore, our research seeks to determine whether enhancing the training data of GAPS with additional geometry maths problems featuring diverse domain-specific languages (DSLs) could lead to performance improvements. Consequently, we conduct experiments to investigate the potential advantages of supplementing the GPAS model's training data with datasets that incorporate various DSLs alongside the UniGeo dataset. Specifically, we extend the data by incorporating the PGPS9K dataset [5], which represents the latest collection of geometry math problems sourced independently from the UniGeo dataset. The PGPS9K dataset consists of a total of 9,022 geometry math problems, out of which 2,891 problems are selected from another dataset called Geometry3K [60]. Notably, the PGPS9K dataset encompasses 30 types of operators in its DSL, whereas the UniGeo dataset comprises 18 types of operators for CAL problems and 44 types of operators for PRV problems. To evaluate the GPAS model's performance, we use two distinct subsets of the PGPS9K test data: the first subset (Geometry3K test) includes 589 problems from the original Geometry3K dataset, and the second subset (PGPS9K test) includes 1000 problems sourced from the PGPS9K dataset.

Training Dataset	CAL (%)	PRV (%)	PGPS9K test (%)	Geometry3K test (%)
UniGeo	67.8	**97.5**	n/a	n/a
PGPS9K	n/a	n/a	50.5	57.5
UniGeo + PGPS9K	**68.5** (*+1.7*)	97.2 (*-0.3*)	**61.2** (*+10.7*)	**68.0** (*+10.5*)

Table 5.4: The performance of top-10 accuracy (%) achieved by GAPS model on various datasets, inlcuding UniGeo CAL, UniGeo PRV, PGPS9K test, and PGPS9K test datasets. The column displaying the highest accuracy is marked in bold. For datasets that were not used during the training stage, the accuracy is not reported, and the corresponding cells are marked as "n/a."

In Table 5.4, we present the varied performances of the GAPS model based on different training approaches. Specifically, we compare the model's performance when trained on three different datasets: UniGeo dataset alone, PGPS9K dataset alone, and the combined UniGeo and PGPS9K datasets. The results reveal a notable improvement in the GAPS model's performance on UniGeo CAL, PGPS9K, and Geometry3K datasets when trained on the combined dataset, in comparison to training with a single dataset. The accuracy shows an increase of 1.7%, 10.7%, and 10.5% on UniGeo CAL, PGPS9K, and Geometry3K, respectively. Despite the substantial differences in the DSL between UniGeo and PGPS9K, the addition of the PGPS9K dataset during training still enhances the GAPS model's capability to solve geometry maths problems. However, it is noteworthy that the GAPS model experiences a negligible decrease in performance on the UniGeo PRV test data when trained on both UniGeo and PGPS9K datasets. This can be attributed to the distinct problem types present in the PGPS9K dataset compared to the UniGeo PRV dataset. The former dataset primarily consists of calculation-type problems, while the latter focuses on proving-type problems.

In summary, the findings from Table 5.4 demonstrate that augmenting the GAPS model's training data with datasets featuring different Domain-Specific Languages (DSLs) can actually enhance its performance, rather than causing any detriment. Interestingly, the design of GAPS, which separates the generation of operators and operands, plays a crucial role in mitigating the complexities introduced by the additional dataset. This design choice allows GAPS to effectively capitalise on the benefits offered by the new dataset without getting overwhelmed by its inherent complexity. As a result, GAPS demonstrates improved performance when trained on the combined dataset with a diverse DSL, underlining the advantages of incorporating data from different domains.

5.5.5 Analysis of the Utility of the Problem-Type Classifier (RQ-5.5, RQ-5.6, RQ-5.7)

This section explores the significance of the problem-type classifier in GAPS, which enables the model to generate solution programs for geometry maths problems belonging to different types simultaneously.

5.5.5.1 The Impact of Problem-Type Classifier on GAPS Performance

The comparison between the performance of GAPS with and without using the problem-type classifier is illustrated in Table 5.5. When GAPS is trained solely on the UniGeo dataset, the presence of the problem-type classifier leads to improved performances.

Train Data	Test Data	w/o (%)	w (%)	three types (%)
UniGeo	CAL	46.8	**67.8**	n/a
	PRV	96.9	**97.5**	n/a
UniGeo + PGPS9K	CAL	40.4	**68.5**	49.4
	PRV	97.1	97.2	**97.9**
	PGPS9K test	53.0	**61.2**	58.8
	Geometry3K test	61.0	**68.0**	63.5

Table 5.5: The comparison between the top-10 accuracy (%) achieved by GAPS model under three configurations: 1. without the problem-type classifier (denoted as "w/o"), 2. equipped with the problem-type classifier (denoted as "w"), 3. considering UniGeo CAL, UniGeo PRV, and PGPS9K as three different problem-types (denoted as "three types"). The highest accuracy scores in each row are highlighted in bold type, indicating the best performance achieved by the respective configuration for each problem type.

Specifically, GAPS equipped with the problem-type classifier achieves a substantial increase of 21% in accuracy on the UniGeo CAL dataset and a 0.6% increase on the UniGeo PRV dataset, compared to GAPS without it. Furthermore, when GAPS is trained on the combined UniGeo and PGPS9K datasets, the importance of the problem-type classifier becomes even more evident. GAPS without the problem-type classifier exhibits notably lower accuracy across all test data in comparison to GAPS with the classifier. These results presented in Table 5.5 emphasise the effectiveness of the problem-type classifier. By acting as a discriminative task, the problem-type classifier efficiently handles the complexity within the generation of solution programs, leading to enhanced performance for the GAPS model on various geometry maths problem types.

5.5.5.2 Evaluating the Influence of Fine-Grained Problem-Type Classification on GAPS Performance

We initiated an inquiry into the potential benefits of categorising geometry maths problems into finer problem types. To explore this, we treated UniGeo CAL, UniGeo PRV, and PGPS9K as three distinct problem types. As depicted in Table 5.5, when dividing the datasets into three problem types, the performance of GAPS exhibited a decline on UniGeo CAL, PGPS9K, and Geometry3K test data, with accuracy drops of 19.1%, 2.3%, and 4.5%, respectively. Upon closer examination, we realised that problems within the UniGeo CAL, PGPS9K, and Geometry3K datasets predominantly focus on calculation problems, warranting their categorisation into a single problem type. Despite the performance decrease in the finer-grained problem type classification, the accuracy of GAPS on these three test datasets remains substantially higher than GAPS without any problem-type classifier at all. From these observations, we can confidently conclude that GAPS indeed benefits from the problem-type classifier, and the extent of its benefits is maximised when the correct problem types are appropriately identified.[2]

5.5.6 GAPS Performance on GeoEval Benchmark

GeoEval Benchmark			
GeoEval-2000 †	GeoEval-backward	GeoEval-aug †	GeoEval-hard
55.1	5.2	26.7	0.0

Table 5.6: Top-10 accuracy (%) of the GAPS model on the GeoEval benchmark. The † denotes that GAPS was evaluated on partial subsets of GeoEval-2000 and GeoEval-aug, as it can only solve geometry math problems with available diagrams. Approximately 62% of problems in the GeoEval-2000 and GeoEval-aug subsets contain diagrams, sourced from the PGPS9K, UniGeo, GeoQA+, and Geometry3K datasets.

Table 5.6 presents the top-10 accuracy performance of the GAPS model on the GeoEval benchmark, which was introduced in Chapter 4. Since GeoEval does not provide annotated solution programs, we employ the GAPS model from Section 5.5.4, trained on a combination of the UniGeo and PGPS9K datasets, to solve the problems in the GeoEval benchmark. Additionally, we compare the execution accuracy, which calculates the accuracy between the golden executable result and the result obtained from the predicted program, because GeoEval only provides the golden executable result, without annotated solution programs.

[2]Please see Appendix C.2 for a further study on how the problem-type classifier accelerates the convergence in GAPS training process.

The results show that GAPS achieved 55.1% accuracy on the GeoEval-2000 subset, despite not being directly trained on this subset. This reasonable performance can be attributed to the fact that we only evaluated GAPS on the partial GeoEval-2000 subset containing both problem texts and geometric diagrams. This partial subset is sourced from the PGPS9K, UniGeo, GeoQA+, and Geometry3K datasets, three of which were used to train GAPS, as described in Section 5.5.4.

However, a concerning observation is that GAPS' performance drops by the 28.4% accuracy on the GeoEval-aug subset compared to the GeoEval-2000 subset. Notably, the problems in GeoEval-aug are rephrased versions of the problems in the GeoEval-2000 subset, with no modifications to the underlying problem statements. This decrease in accuracy demonstrates that GAPS is vulnerable to the semantic diversity introduced by the rephrased problems in the GeoEval-aug subset. Despite being trained on similar geometric concepts, GAPS struggles to generalize its knowledge and reasoning capabilities to linguistically diverse representations of the same underlying problems. This finding highlights a crucial limitation in GAPS's ability to robustly understand and solve geometry problems across various linguistic formulations and phrasings.

The results in Table 5.6 reveal the limitation of the GAPS model when it comes to backward reasoning. GAPS achieves an accuracy of only 5.2% on the GeoEval-backward subset, which is strikingly low. This poor performance in backward reasoning tasks is understandable, given that the GAPS model adopted for this experiment from Section 5.5.4 was not specifically trained to develop backward reasoning capabilities. Backward reasoning in geometry problems involves working backwards from a given conclusion or result to derive the initial conditions or premises that lead to that outcome. This type of reasoning requires a deep understanding of geometric concepts and the ability to reverse the logical flow of problem-solving steps. However, the GAPS model's training focused primarily on forward reasoning, where the model learns to solve problems by applying a sequence of logical steps from given premises to reach a conclusion.

The results in Table 5.6 reveal that the GAPS model is unable to resolve problems in the GeoEval-hard subset, highlighting two critical limitations. Firstly, GAPS was not specifically trained to solve problems related to solid geometry and analytic geometry, which are covered in the GeoEval-hard subset. These advanced geometric domains involve concepts and problem-solving techniques that were not part of the model's training domain. Secondly, the GAPS model adopted from Section 5.5.4 lacks familiarity with the domain-specific language used in the GeoEval-hard subset. This hinders its ability to generate targeted solution programs for the corresponding problems.

5.5.7 Analysis for the Different Cache Token Updating Methods (RQ-5.8)

Table 5.7 presents a performance comparison of various strategies for updating cache tokens. As indicated in the table, employing the query vector used to generate the final operand in the sub-program to update the cache token achieves the highest accuracy across all datasets. Our hypothesis is that this outcome can be attributed to the comprehensive nature of the query vector q_t^{oe}, which encapsulates information from all the symbols generated within the sub-program. Consequently, it can be interpreted as an integrated representation of the sub-program. In contrast, the other two vectors used to update the cache token, q_0^{op} and e_{op}, do not take into account the generated operands, resulting in the incorporation of only partial information from the sub-program.

	CAL (%)	PRV (%)	PGPS9K (%)	Geometry3K (%)
q_0^{op}	59.0	91.0	36.5	40.2
e_{op}	58.8	90.2	37.6	42.2
q_t^{oe}	**65.4**	**97.8**	**50.5**	**57.5**

Table 5.7: Performance comparison between different cache token updating methods. q_0^{op} refers to the query vector used for generating the operator for the sub-program, e_{op} refers to the embedding of the generated operator for the sub-program, and q_t^{oe} refers to the query vector used for generating the last operand for the sub-program. For an equitable evaluation, GAPS models trained exclusively on UniGeo CAL, UniGeo PRV, and PGPS9K datasets are used to assess the performance on their respective test datasets.

5.5.8 Enhancing Solution Program Generation with Hierarchical Beam Search (RQ-5.9)

To enable GAPS to utilise beam search for generating solution programs, we propose the hierarchical beam search, which is designed to be compatible with GAPS' architecture that separates the generation of operators and operands. In order to showcase the necessity and effectiveness of the hierarchical beam search, we conduct a comparison between GAPS employing the hierarchical beam search and GAPS using the traditional greedy decode approach. The results are presented in Table 5.8, which clearly demonstrates that GAPS with hierarchical beam search outperforms the GAPS with greedy decode by a large margin.

	Greedy Decode (%)	Hierarchical Beam Search (%)
CAL	42.1	**65.4** (*+23.3*)
PRV	97.5	**97.8** (*+0.30*)
PGPS9K	32.6	**50.5** (*+17.9*)
Geometry3K	35.8	**57.5** (*+21.7*)

Table 5.8: Comparing Accuracy (%) of GAPS with Hierarchical Beam Search and Greedy Decode for Solution Program Generation. For an equitable evaluation, GAPS models trained exclusively on UniGeo CAL, UniGeo PRV, and PGPS9K datasets are used to assess the performance on their respective test datasets.

Specifically, when utilising the hierarchical beam search for solution program generation, GAPS achieves impressive accuracy improvements of 23.3%, 0.3%, 17.9%, and 21.7% on the UniGeo CAL, PRV, PGPS9K, and Geometry3K test data, respectively. This compelling evidence highlights the efficacy of the hierarchical beam search in allowing GAPS to effectively explore various combinations of operators and operands, ultimately leading to the selection of high-quality solution programs. The hierarchical beam search proves to be a crucial enhancement, enabling GAPS to achieve superior performance in generating accurate and diverse solution programs for geometry maths problems.

5.5.9 Enhancing geometric element Selection with the geometric elements Enhancement Method (RQ-5.10)

The geometric elements enhancement method enhances GAPS' capability to select geometric elements as operands by appending the geometric elements to the problem text. In addition, this enhancement proves particularly valuable in situations where the elements involve identical points (e.g., $\angle SUW$ and $\triangle SUM$), enabling GAPS to differentiate between such elements effectively. To substantiate the effectiveness of this approach, we compare the performances on UniGeo PRV problems between GAPS with the enhancement method and GAPS without it. The results, as presented in Table 5.9, clearly demonstrate the substantial benefits that GAPS gains from the geometric elements enhancement method.[3]

	PRV (%)		CAL (%)	
	Top1	Top10	Top1	Top10
w/o	88.4	89.0	42.1	67.8
w	97.5	97.5	41.6	65.6

Table 5.9: Performance comparison between GAPS with (w) geometric elements enhancement and without (w/o) it.

Similarly, we explore the applicability of a method akin to the geometric elements enhancement to improve GAPS' performance on CAL problems. Since numbers in the text are not tokenized into unknown tokens, we solely insert the token "number" before each number and append all numbers to the problem text. However, as demonstrated in Table 5.9, contrary to the results observed with the geometric elements enhancement, GAPS' performances decline when employing this method. The findings in Table 5.9 indicate that the approach of inserting "number" tokens and appending the numbers to the problem text does not yield the desired performance improvements for GAPS on CAL problems. Consequently, this method does not prove as effective in enhancing GAPS' ability to handle CAL problem types as compared to its success with PRV problems.

[3] Please see Appendix C.3 for a case study where we map out the probability distributions during the generation of operands.

5.6 Chapter Summary

In Chapter 5, we focused on adapting our proposed methods to solve math problems with the additional component of geometric diagrams. This chapter states that we could effectively leverage the unique features of various geometry problem types without encountering a decline. In line with this statement, we propose the Geometry-Aware Problem Solver (GAPS), which can simultaneously handle diverse types of geometry math problems. GAPS achieves this by integrating a problem-type classifier, enabling a unified solution program generator to cater to various problem types. This allows GAPS to independently generate operators and operands using symbols from domain-specific languages tailored to each problem type. The problem-type classifier also enables GAPS to incorporate diverse data types for supplementary training without compromising performance.

To validate our statement, we addressed the research questions outlined in Section 5.1. In Section 5.5.1, we answered research question **RQ-5.1** by comparing GAPS with leading methods on two types of geometry problems: calculation and proving. As shown in Table 5.2, GAPS outperformed the baseline models.

In Section 5.5.2, we compared GAPS's accuracy with other baseline models based on sub-program types. Figure 5.4 illustrates that GAPS excelled in both main problem types and their sub-tasks, addressing research question **RQ-5.2**.

We observed a notable disparity in GAPS's performance on proving versus calculation problems, addressing research question **RQ-5.3**. Section 5.5.3 reveals that most errors in calculation problems stem from incorrect operators, while proving problems typically suffer from incorrect operands. This suggests that calculation problems present a more diverse and challenging set of solution programs. Consequently, GAPS performs better on proving problems than on calculation problems, thus addressing **RQ-5.3**.

We further investigated the impact of training on diverse geometry math problems with different domain-specific languages. In Section 5.5.4, we incorporated the PGPS9K dataset, comprising newly created problems and those from the Geometry3K dataset. We assessed the model's performance using three datasets: solely UniGeo, solely PGPS9K, and a combination of both. The results, shown in Table 5.4, indicate performance improvements when GAPS was trained on the combined dataset, confirming **RQ-5.4**.

In Section 5.5.5, we explored the importance of the problem-type classifier, addressing research questions **RQ-5.5**, **RQ-5.6**, and **RQ-5.7**. Our ablation study, as shown in Table 5.5, revealed that including the problem-type classifier enhances GAPS's performance on both the UniGeo and combined UniGeo-PGPS9K datasets. Its absence led to a marked decrease in accuracy. Optimizing the number of problem types maximized GAPS's benefits. Additionally, training loss convergence analysis indicated faster convergence for GAPS with the problem-type classifier, confirming its effectiveness.

We addressed research question **RQ-5.8** by analyzing the effectiveness of updating the cache token with the query vector used for generating the final operand in the sub-program, as detailed in Table 5.7. This approach yielded the highest accuracy, validating our method. Additionally, we developed a specialized algorithm for hierarchical beam search due to our unique design that separates operator and operand generation. The results in Table 5.8 demonstrate that employing beam search enhances GAPS's accuracy, effectively answering research question **RQ-5.9**.

To address research question **RQ-5.10**, we performed an ablation study by removing the geometry elements enhancement feature and evaluated its impact on the UniGeo dataset. The results, shown in Table 5.9, highlight the advantages of incorporating this method. A case study visualizing the probability distributions during operand generation with and without this enhancement, depicted in Figure C.2, indicates that GAPS is more likely to select the correct geometry elements when the enhancement is applied.

Overall, our findings provide strong evidence of GAPS' efficacy and highlight the importance of incorporating advanced techniques to enhance its performance on geometry problem-solving tasks. However, we discovered that vector-based representations of geometric relations struggle to accurately capture the complex relationships within geometric diagrams. Additionally, these representations are not easily interpretable by humans, making it challenging to identify whether performance issues stem from the relation extraction or the problem-solving component. In Chapter 6, we propose a new representation method to describe geometric diagrams, aiming to enhance interpretability and generalization ability.

Chapter 6

GOLD: Geometry Problem Solvers with Natural Language Description

Chapter 5 presents a novel model designed to comprehend and solve geometry maths problems by integrating various modalities, namely geometric diagrams and problem texts. The approach primarily focuses on jointly encoding these diagrams and texts into high-dimensional vectors, aiming to create a cohesive understanding of both elements. Despite this innovative approach, the model faces challenges in accurately interpreting geometric diagrams, especially due to the intricate nature of geometric primitives like lines and arcs, which are often slender and overlapped. This complexity necessitates a more detailed description of the relationships among diagram components, a task at which the model introduced in Chapter 5 still encounters difficulties, resulting in limited interpretability.

To address this, in this Chapter we present the **G**eometry problem s**O**lver with natural **L**anguage **D**escription (GOLD) model. GOLD enhances the extraction of geometric relations by separately processing symbols and geometric primitives within the diagram. Subsequently, it converts the extracted relations into natural language descriptions, efficiently utilising large language models to solve geometry maths problems.Experiments show that the GOLD model outperforms the Geoformer model, the previous best method on the UniGeo dataset, by achieving accuracy improvements of 12.7% and 42.1% in calculation and proving subsets. Additionally, it surpasses the former best model on the PGPS9K and Geometry3K datasets, PGPSNet, by obtaining accuracy enhancements of 1.8% and 3.2%, respectively. This chapter is mainly based on my published paper “GOLD: Geometry Problem Solver with Natural Language Description” accepted in 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024 Findings).

Section 6.1 delves into a categorisation of current methodologies, specifically distinguishing between symbolic and neural solvers, and discusses their respective strengths and limitations. It also outlines the research questions addressed and the contributions made in this chapter. In Section 6.2, we offer a detailed explanation of our approach, particularly how we utilise natural language descriptions to represent geometric diagrams and apply these descriptions to solve geometry maths problems. Section 6.3 outlines the datasets and metrics employed in our evaluation. Section 6.4 is dedicated to a thorough analysis of the experimental results. The chapter concludes with Section 6.5, which provides a summary of the work accomplished in this chapter.

6.1 Introduction

As discussed in Section 2.3 and Section 5.1, automated solving of geometry maths problems has gained considerable attention in the AI community in recent years. Unlike maths word problems, geometry maths problems involve additional geometric diagrams, necessitating comprehensive reasoning capabilities for understanding multi-modal information. As a result, research on the automated resolution of geometry problems is still in its infancy [1].

Existing approaches for solving geometry maths problems utilise neural networks to embed the diagram and problem text separately or jointly. For example, the GAPS model proposed in Chapter 5 utilise neural networks to embed the diagram and problem text jointly, resulting in highly generalised models. However, these methods struggle with accurately capturing the complex relationships within geometric diagrams [20]. Additionally, their vector-based representation of geometric relations is not easily interpretable by humans, posing challenges in identifying whether performance issues are from the relation extraction or the problem-solving component. In a different approach, some studies have successfully translated geometric diagrams into formal languages, enhancing precision and interpretability [43, 45]. However, these methods do not separately process relations among geometric primitives and relations between symbols and geometric primitives, which adds difficulty in solving the geometry maths problem correctly. Moreover, these approaches necessitate specifically designed solvers that take formal languages as input, making them incompatible with prevalent large language models (LLMs).

To address the limitations of existing methods in solving geometry maths problems, this chapter states that using natural language descriptions for geometric diagrams can provide accurate and understandable representations while bridging the gap between textual and visual information (See Statement (3) in Section 1.3). Therefore, we introduce the GOLD model in this chapter. The GOLD model converts geometric diagrams into natural language descriptions, aiding in the generation of solution programs for the problems. Particularly, the GOLD model's relation-construction head extracts two types of geometric relations: *sym2geo* (relations between symbols and geometric primitives) and *geo2geo* (relations among geometric primitives). This process involves two specialised heads that separately model symbols and geometric primitives within diagrams as distinct vectors. These extracted geometric relations are then converted into natural language descriptions. This not only improves the model's interpretability but also connects geometric diagrams with problem texts. Furthermore, since these natural language descriptions meet the input requirements of LLMs, the GOLD model is able to utilise the advanced LLMs as the problem-solving module, efficiently generating solution programs used to solve geometry maths problems.

To validate the statement, this Chapter investigates the following research questions (RQ):

- **RQ-6.1 Performance and Accuracy**: In comparison with existing approaches like Geoformer and PGPSNet, how are the accuracy improvements achieved by the GOLD model across different datasets such as UniGeo, PGPS9K, and Geometry3K?

- **RQ-6.2 Natural Language Representation**: How does converting extracted geometric relations into natural language enhance the ability of the model to generate solution programs, and how does this natural language representation compare with the formal language used by symbolic solvers in terms of alignment with current language models?
- **RQ-6.3 The geo2geo and sym2geo Relation Extraction Accuracy**: How does different embeddings influence the geo2geo and sym2geo relation extraction accuracy?
- **RQ-6.4 Influence of feature_embedding and spatial_embedding on Geometry Problem Solving**: How does feature_embedding and spatial_embedding influence the geometry-problem solving?
- **RQ-6.5 Usefulness of the Additional geo2geo Relation**: How does providing additional description of relations between geometry primitives help to solve the geometry maths problems?

6.1.1 Contributions

To summarise, the contributions of this chapter are:

1. We propose the GOLD model to extract geometric relations from geometric diagrams and subsequently convert these relations into natural languages, which are then utilised for solving geometry maths problems. Its compatibility with LLMs is a significant advantage, enabling the GOLD model to utilise the capabilities of LLMs to generate solution programs.
2. The GOLD model separately processes symbols and geometric primitives from the diagrams. This separation design simplifies the extraction of the geometric relations.
3. Our GOLD model demonstrates improvements over previous methods across all evaluated datasets, validating the effectiveness of our approach.

6.2 Approach

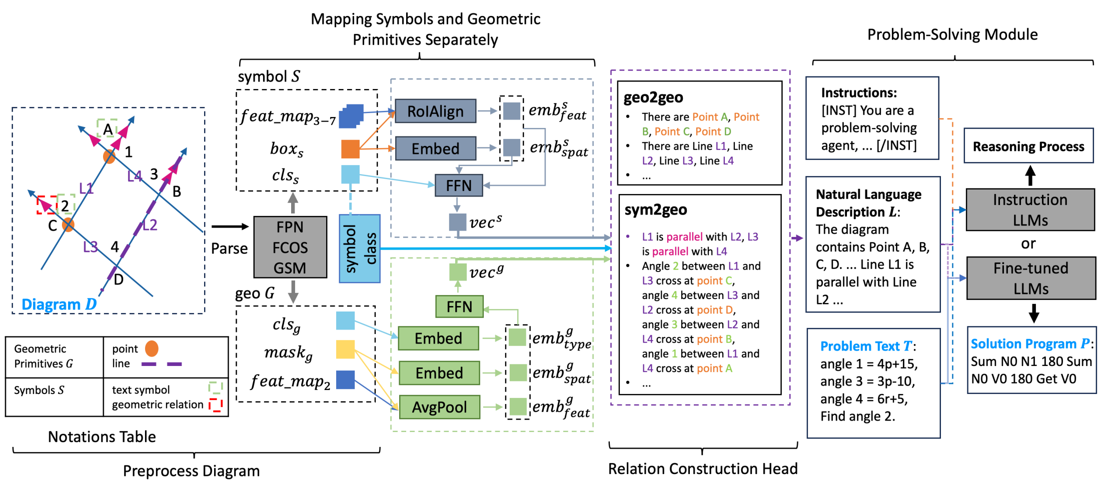

Mapping Symbols and Geometric Primitives Separately
Problem-Solving Module
symbol S
$feat\_map_{3-7}$
box_s
cls_s
RoIAlign
Embed
FFN
emb^s_{feat}
emb^s_{spat}
vec^s
FPN FCOS GSM
symbol class
Parse
geo G
cls_g
$mask_g$
$feat\_map_2$
vec^g
FFN
Embed
Embed
AvgPool
emb^g_{type}
emb^g_{spat}
emb^g_{feat}
A
B
C
D
1
2
3
4
L1
L2
L3
L4
Diagram D
Geometric Primitives G
point
line
Symbols S
text symbol
geometric relation
Notations Table
Preprocess Diagram
geo2geo
There are Point A, Point B, Point C, Point D
There are Line L1, Line L2, Line L3, Line L4
...
sym2geo
L1 is parallel with L2, L3 is parallel with L4
Angle 2 between L1 and L3 cross at point C, angle 4 between L3 and L2 cross at point D, angle 3 between L2 and L4 cross at point B, angle 1 between L1 and L4 cross at point A
...
Relation Construction Head
Instructions: [INST] You are a problem-solving agent, ... [/INST]
Natural Language Description L: The diagram contains Point A, B, C, D. ... Line L1 is parallel with Line L2 ...
Problem Text T: angle 1 = 4p+15, angle 3 = 3p-10, angle 4 = 6r+5, Find angle 2.
Reasoning Process
Instruction LLMs
or
Fine-tuned LLMs
Solution Program P: Sum N0 N1 180 Sum N0 V0 180 Get V0

Figure 6.1: The illustration of the GOLD Model. The diagram $\mathcal{D}$, problem text $\mathcal{T}$, and solution program $\mathcal{P}$ used in this illustration are sourced from the PGPS9K dataset [5]. The symbols and geometric primitives in the diagram are annotated using the notations from the Notation Table, which are consistent with the colours of extracted relations of sym2geo and geo2geo.

Our GOLD model is illustrated in Figure 6.1.

6.2.1 Task Description and Pre-parsing

The objective is to generate the correct solution program $\mathcal{P}$ to solve the problem by analysing a geometry maths problem text $\mathcal{T}$ and its corresponding diagram $\mathcal{D}$. Specifically, the solution program represents intermediate steps in the domain-specific language generating the output for the question (see an example of solution program in Figure 6.1).

In our approach, we initially pre-process geometric diagrams to extract geometric primitives $\mathcal{G}$ (including *Point* **P**, *Line* **L**, and *Circle* **C**) and symbols $\mathcal{S}$ from the diagram $\mathcal{D}$ for subsequent task. Specifically, we utilise a standard Feature Pyramid Network (FPN) [223] integrated with a MobileNetV2 [224] backbone for this task. For the detection of symbols, we apply the anchor-free detection model FCOS [8], and for the extraction of geometric primitives, we use the GSM model [183]. The FCOS model employs feature maps P3 to P7, generated by the FPN layer, to detect symbols within the diagram. This detection step produces bounding box coordinates (box_s) and class type (cls_s) for each symbol ($s \in \mathcal{S}$). For the extraction of geometric primitives, we prefer using the feature map P2 instead of P1, as P2 is more memory-efficient due to its lower resolution. This process results in the identification of segmentation masks (mask_g) and class type (cls_g) for each geometric primitive ($g \in \mathcal{G}$).[1]

6.2.2 Mapping Symbols and Geometric Primitives Separately

Before constructing the geometric relations, we map the symbols and geometric primitives into vectors. To achieve this, we introduce two heads: symbol vector head and geometric primitive vector head. Specifically, each head functions as extracting the *feature_embedding* (emb_{feat}) and *spatial_embedding* (emb_{spat}). The *feature_embedding* is computed from the cropped feature map, which is determined by either the bounding box or the segmentation mask. Moreover, where symbols and geometric primitives are placed significantly shapes how they relate. For instance, only points lying on a line can hold the geometric relation with that particular line. Thus, we hypothesise that incorporating spatial information of $\mathcal{S}$ and $\mathcal{G}$ can enhance the accuracy of predictions about geometric relations. Consequently, we embed the bounding boxes of symbols and the coordinates of the geometric primitives into the *spatial_embedding*.

6.2.2.1 Constructing the *feature_embedding*

To obtain the *feature_embedding* ($\text{emb}^{s,g}_{feat}$) and *spatial_embedding* ($\text{emb}^{s,g}_{spat}$) for symbol s or geometric primitive g, we conduct the below calculation:

$$\text{emb}^{s,g}_{feat} = \text{ReLU}(\mathbf{W}^{s,g}_{feat}\mathbf{V}^{s,g}) \tag{6.1}$$

where $\mathbf{W}^{s,g}_{feat} \in \mathbb{R}^{h \times h}$ are trainable parameters for either symbols or geometric primitives. Next, we elaborate the calculation process of $\mathbf{V}^{s,g}$ for symbols and geometric primitives separately.

[1]The knowledge of FCOS and GSM models are not this thesis's work, however, they are the foundational knowledge required for understanding this Section. Therefore, we put them in the Appendix D.1.

To obtain the $\mathbf{V}^s$ for symbol s, we utilise RoIAlign [193] on its feature map, based on the bounding box of symbol s:

$$\mathbf{V}^s = \mathbb{F}(\mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv}(\mathrm{RoIAlign}(\mathrm{box}_s, \mathrm{feat\_map}_i))))) \tag{6.2}$$

where i refers to the i-th layer of feature maps where the bounding box (box_s) is calculated from. The Conv is the convolution layer with 64 channels, BN is the BatchNorm layer, and ReLU is the ReLU activation layer. The $\mathbb{F}$ means flatten operation, indicating that the $\mathbf{V}^s$ is further flatten into a vector and used for obtaining the *feature_embedding* emb^s_{feat} for symbol s through Eq 6.1.

To obtain the $\mathbf{V}^g$ for geometric primitive g, we perform an element-wise multiplication between the segmentation mask (mask_g) of g and the P2 layer of feature map ($\mathrm{feat\_map}_2$). Next, we flatten the resulting vector along the height and width dimensions and apply global average pooling to obtain the $\mathbf{V}^g$:

$$\mathbf{V}^g = \mathrm{AvgPool}(\mathbb{F}(\mathrm{mask}_g \times \mathrm{feat\_map}_2)) \tag{6.3}$$

The $\mathbf{V}^g$ is used for calculating the *feature_embedding* emb^g_{feat} for geometric primitive g through Eq 6.1.

6.2.2.2 Constructing the *spatial_embedding*

The *spatial_embedding* is obtained by mapping the spatial information of symbols and geometric primitives into embeddings. Specifically, for symbol s, we map the coordinates of its bounding box into an embedding using the trainable parameters $\mathbf{W}^s_{spat} \in \mathbb{R}^{h\times 4}$. Specifically, $\mathrm{emb}^s_{spat} = \mathbf{W}^s_{spat}[x_t, y_t, x_b, y_b]^\top$, where (x_t, y_t) represent the coordinates of the top-left corner of the bounding box, and (x_b, y_b) is the coordinates of the bottom-right corner of the bounding box.

Next, to obtain the *spatial_embedding* of a geometric primitive g, we start by representing coordinates of g using loc_g. The format of loc_g depends on the class type (cls_g) of the geometric primitive: for a point, it contains two numbers ($n_g = 2$) representing its coordinates; for a line, it contains four numbers ($n_g = 4$) representing the coordinates of its start and end points; and for a circle, it contains three numbers ($n_g = 3$)

representing the coordinates of its centre point and the radius length. We then map loc_g into *spatial_embedding* by calculating $\text{emb}^g_{spat} = \text{ReLU}(\mathbf{W}^g_{spat}(\mathbf{W}^g_{loc} loc_g))$, where $\mathbf{W}^g_{loc} \in \mathbb{R}^{h \times n_g}$ are different trainable parameters for different cls_g, and $\mathbf{W}^g_{spat} \in \mathbb{R}^{h \times h}$ are trainable parameters.

To help the model differentiate between different types of geometric primitives, we introduce the *geo_type_embedding* (emb^g_{type}) to capture the semantic information of the geometric primitive. The emb^g_{type} is obtained by performing a lookup operation on the embeddings using the class type (cls_g) of g from the list of geometric primitive types $[\mathbf{P}, \mathbf{L}, \mathbf{C}]$. Specifically, $\text{emb}^g_{type} = \text{embedding}(\text{cls}_g)$, where cls_g is the class type ID of g.

6.2.2.3 Symbol Vector and Geometric Primitive Vector

The vector representation $\text{vec}^{s \in \mathcal{S}}$ of symbol s is obtained by passing concatenated emb^s_{feat} and emb^s_{spat} through a specific feed-forward neural network:

$$\text{vec}^{s \in \mathcal{S}} = \text{ReLU}(\mathbf{W}^s_{vec}[\text{emb}^s_{feat} : \text{emb}^s_{spat}]^\top) \tag{6.4}$$

where $\mathbf{W}^s_{vec} \in \mathbb{R}^{h \times 2h}$ are the trainable parameters depending on the class type (cls_s) of symbol s, and [:] refers to concatenation operation.

The vector representation of the geometric primitive $\text{vec}^{g \in \mathcal{G}}$ is obtained by summing up three embeddings relevant to the geometric primitive g, emb^g_{feat}, emb^g_{spat}, and emb^g_{type}:

$$\text{vec}^{g \in \mathcal{G}} = \text{ReLU}(\mathbf{W}^g_{vec}(\text{emb}^g_{feat} + \text{emb}^g_{spat} + \text{emb}^g_{type})) \tag{6.5}$$

where $\mathbf{W}^g_{vec} \in \mathbb{R}^{h \times h}$ are the trainable parameters.

6.2.3 Relation Construction Head

The relation-construction head aims to establish *sym2geo* relations among symbols and geometric primitives and *geo2geo* relations among geometric primitives themselves.

6.2.3.1 sym2geo relation

The *sym2geo* relation can be further divided into *text2geo* and *other2geo* relations. The *text2geo* relation explains the association between text symbols and geometric primitives, where the text symbols are used to be the reference to a geometric primitive or to display degree, length, etc. To distinguish the role of a text symbol, we introduce the *text_class* for the text symbol. Specifically, when *text_class* is category *0*, the *text2geo* signifies point (or line, or circle) names; when *text_class* is category *1*, the *text2geo* corresponds to angle degrees; when *text_class* is category *2*, the *text2geo* signifies line lengths; when *text_class* is category *3*, the *text2geo* denotes the degree of an angle within a circle. The probabilities of the category ($P(\textit{text\_class}|s)$) of text symbol ($s \in \{\mathcal{S}|\text{cls}_s = \text{“text”}\}$) is defined as:

$$P = \text{softmax}(\mathbf{W}^{sym2geo}_{text\_class}\text{ReLU}(\mathbf{W}^{sym2geo}_{1}\text{vec}^s)) \tag{6.6}$$

where $\mathbf{W}^{sym2geo}_{1} \in \mathbb{R}^{h\times h}$ and $\mathbf{W}^{sym2geo}_{text\_class} \in \mathbb{R}^{4\times h}$, both are the trainable parameters.

The *other2geo* relation captures relations between non-text symbols ($s \in \{\mathcal{S}|\text{cls}_s \neq$ “text”$\}$) and geometric primitives. The non-text symbols are used to find out the relations among geometric primitives, such as *angles of same degree*, *lines of same length*, *parallel lines*, and *perpendicular lines*. For instance, in Figure 6.1, the symbol enclosed in a red rectangle signifies the parallel relation.

To establish the *sym2geo* relation between symbol s and geometric primitive g, we begin by utilising the corresponding symbol head to transform the vector of the geometric primitive: $\widehat{\text{vec}}^g$=ReLU($\mathbf{W}^{sym2geo}_{s1}\text{vec}^g$), where $\mathbf{W}^{sym2geo}_{s1} \in \mathbb{R}^{h\times h}$ are trainable parameters that vary depending on different class types (cls$_s$) of symbols. Finally, we calculate the probabilities of the existence of the relation between symbol s and geometric primitive g as follows:

$$\begin{aligned} O_1 &= \text{ReLU}(\mathbf{W}^{sym2geo}_{2}[\text{vec}^s : \widehat{\text{vec}}^{g\in\{sub\}}]) \\ P(\text{rel}^{sym2geo}_{s,g}|s,g) &= \text{sigmoid}(\mathbf{W}^{sym2geo}_{rel}O_1) \end{aligned} \tag{6.7}$$

where $\mathbf{W}^{sym2geo}_{2} \in \mathbb{R}^{h\times 2h}$ and $\mathbf{W}^{sym2geo}_{rel} \in \mathbb{R}^{1\times h}$ are the trainable parameters. Worth mentioning, that each type of symbol, including the additional four categories of the text symbol, has its own $\mathbf{W}_2^{sym2geo}$. Additionally, $\{sub\}$ refers to the subset of geometric primitives, as certain symbols can only have relations with specific geometric primitives.

6.2.3.2 geo2geo relation

Previous work tend to provide only *sym2geo* relations. However, despite the *sym2geo* relation can provide geometric relations among geometric primitives like parallel, perpendicular, etc. We hypothesise that providing additional information that describes all the geometric primitives from the diagrams is beneficial for the task. Moreover, we tackle the issue concerning the absence of references to geometric primitives in the diagram. For example, in Figure 6.1, the original diagram lacks a reference to the line, where *sym2geo* relation cannot address. To overcome this limitation, we have devised an automated approach that assigns appropriate references to the geometric primitives using the format "cls_g + num" (e.g., "L1, L2, L3, L4" in purple in Figure 6.1). This enables the relation-construction module to (1) present a detailed depiction of the diagram by describing the *geo2geo* relations, even in the absence of a single reference, and (2) generate all *sym2geo* relations, even when some geometric primitives lack references. The *geo2geo* relations are categorised according to the involved geometric primitives: (1) Point and Line: "on-a-line" and "end-point". The "on-a-line" relation occurs when a point lies between the tail and the head of the line. Specifically, a point lying at either the head or the tail of the line is the "end-point", which is the special case of "on-a-line". (2) Point and Circle: "centre-point" and "on-a-circle." The "centre-point" relation refers to a point being the centre point of the circle. The "on-a-circle" relation occurs when a point lies on the arc of the circle. Finally, the probabilities ($P(\text{rel}_{g_i,g_j}^{geo2geo}|g_i, g_j)$) of the relations between geometric primitives g_i and g_j can be calculated as follows:

$$P = \text{softmax}(\mathbf{W}_{rel}^{geo2geo}\text{ReLU}(\mathbf{W}_1^{geo2geo}(\text{vec}^{g_i} + \text{vec}^{g_j}))) \tag{6.8}$$

where $\mathbf{W}_1^{geo2geo} \in \mathbb{R}^{h\times h}$ and $\mathbf{W}_{rel}^{geo2geo} \in \mathbb{R}^{3\times h}$ are the trainable parameters (the number 3 refers to "no relation" and two relations from either Point and Line or Point and Circle).

6.2.4 Problem-Solving Module

Both the *sym2geo* and *geo2geo* relations are expressed in natural languages by the GOLD model, following the same format as the problem text $\mathcal{T}$ relations to natural language descriptions.[2] Therefore, it is convenient to utilise the LLMs as the problem-solving module. Specifically, the problem text $\mathcal{T}$ and the natural language descriptions $\mathcal{L}$ are concatenated for the LLMs to generate the solution program $\mathcal{P}$. To illustrate the compatibility of our methods with LLMs, we employ three well-known models for problem-solving: T5-base [87], Llama2-13b-chat [75], and CodeLlama-13b [225]. The T5-base model is fine-tuned for the target solution programs. Conversely, for Llama2-13b-chat and CodeLlama-13b, we employ directive instructions to guide their solution generation process.[3]

6.2.5 Training Objective

Given a dataset of geometry maths problems. The training process begins with training the pre-parsing module to extract necessary features from the geometric diagrams. Following this, we focus on training three components: the symbol vector head, the geometric primitive vector head, and the relation-construction head. This training is guided by minimising a joint loss function, which is defined as $L_{cons} = L_{g2g} + L_{t\_cls} + L_{s2g}$. The L_{g2g} loss represents the negative log-likelihood loss for accurately identifying the ground truth *geo2geo* relations. Meanwhile, the L_{t_cls} constitutes the negative log-likelihood loss for correctly categorising the text symbols. Lastly, the L_{s2g} loss is the binary cross-entropy loss associated with the ground truth *sym2geo* relations. Once they are trained, and their parameters are fixed, we advance to the final stage of fine-tuning the problem-solving module.[4] During this stage, our objective is to minimise the L_{prog} loss, which is the negative log-likelihood loss for correct solution programs

The L_{g2g} is defined as the negative log-likelihood loss, where we aim to minimise

[2]Please see Appendix D.2 for the defined paradigm used to convert *geo2geo* and *sym2geo* relations to natural language descriptions.

[3]Please see Appendix D.3 for the instruction choice.

[4]Note that the fine-tuning step is only implemented when T5-base is used as the problem-solving module.

the negative log-likelihood of the ground truth relations among geometric primitives:

$$L_{g2g} = -\sum_{g_i \in \mathbf{P}} \sum_{g_j \in \mathbf{L},\mathbf{C}} \log(P(\mathrm{rel}_{g_i,g_j}^{geo2geo} | g_i, g_j)) \tag{6.9}$$

where g_i is a geometric primitive belonging to points, and g_j is a geometric primitive belonging to lines and circles. The $\mathrm{rel}_{g_i,g_j}^{geo2geo}$ refers to the ground truth relation between g_i and g_j.

The L_{t_cls} is defined as the negative log-likelihood loss, where we aim to minimise the negative log-likelihood of the ground truth *text_class* of the text symbol:

$$L_{t\_cls} = -\sum_{S} \log(P(text\_class_s | s)) \tag{6.10}$$

where $\mathrm{text\_class}_s$ is the ground truth *text_class* of the symbol s.

The L_{s2g} is the binary cross-entropy loss:

$$\begin{aligned} L_{s2g} = &-\sum_{s \in \mathcal{S}} \sum_{g \in \mathcal{G}} \{\mathbb{I}(s,g) \times \log(P(\mathrm{rel}_{s,g}^{sym2geo} | s, g)) \\ &+ (1 - \mathbb{I}(s,g)) \times (1 - \log(P(\mathrm{rel}_{s,g}^{sym2geo} | s, g)\} \end{aligned} \tag{6.11}$$

where $\mathbb{I}(s,g)$ is 1 if there is relation between symbol s and geometric primitive g, otherwise it is 0.

The L_{prog} is defined as the negative log-likelihood loss, where we aim to minimise the negative log-likelihood of the tokens of the ground truth solution programs:

$$L_{prog} = -\sum_{i} \log(P(t_i | t_{<i})) \tag{6.12}$$

where i is the i-th token in the ground truth solution program.

6.2.6 Inference Stage

During the inference stage, we employ Eq 6.4 and Eq 6.5 to map symbols $s \in \mathcal{S}$ and geometric primitives $g \in \mathcal{G}$ to corresponding vectors $\mathrm{vec}^{s \in \mathcal{S}}$ and $\mathrm{vec}^{g \in \mathcal{G}}$, respectively. Following this, we proceed with the inference of *sym2geo* and *geo2geo* relations.

6.2.6.1 Predict sym2geo Relation

For a text symbol $s \in \{\mathcal{S} | \mathrm{cls}_s = \text{“text”}\}$, it is necessary to determine its meaning based on its *text_class*. To accomplish this, we assign the category of text symbol

$s \in \{\mathcal{S}|\text{cls}_s = \text{“text”}\}$ as the one with the highest probability among the $P(\textit{text\_class}|s)$ values, as specified in Eq 6.6:

$$\textit{text\_class}_s = \textit{argmax} P(\textit{text\_class}|s) \tag{6.13}$$

- if $\textit{text\_class}_s$ is 0 (i.e., category *0*), it indicates that the symbol s corresponds to the reference name of a point, or a line, or a circle. In this case, we assign the symbol s to the geometric primitive g that has the highest probability of $P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P},\mathbf{L},\mathbf{C}\}}|s,g)$, where $g \in \{\mathbf{P},\mathbf{L},\mathbf{C}\}$ specifies that the geometric primitive g belongs to the set of points, lines, and circles:

$$g = \textit{argmax} P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P},\mathbf{L},\mathbf{C}\}}|s,g) \tag{6.14}$$

- if $\textit{text\_class}_s$ is 1 (i.e., category *1*), it indicates that the symbol s represents the degree of an angle. Since an angle consists of two lines and one point, we select the point with the highest probability $P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P}\}}|s,g)$, and we select the two lines with the top two highest probabilities $P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{L}\}}|s,g)$. It is worth mentioning these two lines must have *geo2geo* relations of “end-point” or “on-a-line” with the selected point.

$$\begin{aligned} &p = \textit{argmax} P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P}\}}|s,g) \\ &l_1, l_2 = \textit{argmax}_{\textit{two}} P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{L}\}}|s,g), \\ &\text{where } \textit{rel}_{l_1,p} \in \{\text{“end-point”‘, “on-a-line”}\} \text{ and} \\ &\textit{rel}_{l_2,p} \in \{\text{“end-point”‘, “on-a-line”}\} \end{aligned} \tag{6.15}$$

- if $\textit{text\_class}_s$ is 2 (i.e., category *2*), it indicates that the symbol s represents the length of a line. Since a line consists of two points, we select the points with the top two highest probabilities $P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P}\}}|s,g)$:

$$p_1, p_2 = \textit{argmax}_{\textit{two}} P(\text{rel}^{\textit{sym2geo}}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P}\}}|s,g) \tag{6.16}$$

- if $\textit{text\_class}_s$ is 3 (i.e., category *3*), it indicates that the symbol s represents the degree of an angle on the circle. In this case, the angle is formed by the centre point of a circle and two points lying on the arc of a circle. Therefore, we first select the circle with the highest probability of $P(\text{rel}^{sym2geo}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{C}\}}|s,g)$. Subsequently, we select two points with the top two highest probabilities, which are $P(\text{rel}^{sym2geo}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P}\}}|s,g)$. Worth mentioning, these two points must be on the arc of the selected circle:

$$\begin{aligned}
&c = argmaxP(\text{rel}^{sym2geo}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{C}\}}|s,g)\\
&p_1,p_2 = argmax_{two}P(\text{rel}^{sym2geo}_{s\in\{\mathcal{S}|\text{cls}_s=\text{“text”}\},g\in\{\mathbf{P}\}}|s,g),\\
&\text{where } rel_{p_1,c} = rel_{p_2,c} = \text{“on-a-circle”}
\end{aligned} \tag{6.17}$$

For the geometric relations among geometric primitives, such as parallel. It is determined by the *other2geo* relation. For the *other2geo* relation involving other symbols, it is required that the relation holds with at least two geometric primitives. This means that there should be at least two geometric primitives with probabilities $P(\text{rel}^{sym2geo}_{s,g}|s,g)$ larger than a threshold θ. In this case, the geometric primitives are selected based on this criterion.

$$\begin{aligned}
&\{g_{\text{indices}}\} = \text{sorted}(P(\text{rel}^{sym2geo}_{s\in\{\mathcal{S}|\text{cls}_s\neq\text{“text”}\},g\in\{\mathbf{P},\mathbf{L},\mathbf{C}\}}|s,g)) > \theta\\
&g_{\text{selected}} = \mathcal{G}[\{g_{\text{indices}}\}]
\end{aligned} \tag{6.18}$$

where “sorted” indicates that values are sorted in descending order, and [] refers to the selection from the geometric primitives group $\mathcal{G}$ according to the indices $\{g_{\text{indices}}\}$. The threshold θ is set as 0.5 experimentally.

6.2.6.2 Predict geo2geo Relation

The *geo2geo* relation between geometric primitives g_i and g_j is determined based on Eq 6.8, where it is assigned as the relation with the highest probability:

$$rel_{g_i,g_j\in\mathcal{G}} = argmaxP(\text{rel}^{geo2geo}_{g_i,g_j}|g_i,g_j) \tag{6.19}$$

In an ideal scenario, the OCR results would accurately provide references to the points, lines, and circles, allowing us to extract precise information about the geometric primitives. However, the open-source OCR tool[5] we have adopted is not accurate. As a result, some primitives may lack reference names. To address this issue, we

[5]https://github.com/JaidedAI/EasyOCR

automatically label the primitives in sequential order (e.g., "P1, P2, L1, L2") if their reference names are missing.

6.2.6.3 Generate Solution Program

Once the *geo2geo* and *sym2geo* relations are constructed, we proceed to convert them into natural language descriptions $\mathcal{L}$. We then concatenate the natural language descriptions $\mathcal{L}$ with the problem text $\mathcal{T}$. This combined text is passed to the problem-solving module, which employs BeamSearch with a beam size of 10 to generate the solution program $\mathcal{P}$. Moreover, when using larger LLMs, such as Llama2, we add instructions in front of the concatenation of $\mathcal{L}$ and $\mathcal{T}$, which is further sent to LLMs to generate reasoning process.

6.3 Experimental Setup

Our method was implemented using the PyTorch [203] and HuggingFace [204] libraries. For the pre-parsing module, we followed the training and parameter settings of the previous work [183]. We evaluated the dimensions of the embeddings over a range of {32, 64, 128}, and based on the model's performance in the validation set, we experimentally determined 64 as the optimal dimension size for the embeddings. We utilised the Adam optimiser with a learning rate of $1e^{-4}$ and weight decay of $1e^{-4}$ for training all modules. The symbol vector head, geometric primitive vector head, and relation-construction head were trained end-to-end for 50 epochs with a batch size of 20, while the problem-solving module (using T5-base) was fine-tuned for 30 epochs with a batch size of 10. All experiments were conducted on one NVIDIA A100 80GB GPU.

6.3.1 Datasets

Our experiments are conducted on three datasets: UniGeo [1], PGPS9K [5], and Geometry3K [60]. The UniGeo dataset comprises 14,541 problems, categorized into 4,998 calculation problems (CAL) and 9,543 proving problems (PRV), which are split into train, validate, and test subsets in a ratio of 7.0: 1.5: 1.5. The Geometry3K includes

3,002 problems, divided into train, validate, and test subsets following a 7.0: 1.0: 2.0 ratio. Since PGPS9K contains a partial Geometry3K dataset, we keep an exclusive set of 6,131 problems, of which 1000 problems are a test subset. Due to the absence of a validation subset in PGPS9K, we divide its training set to create a train-validation split in a 9.0: 1.0 ratio.

6.3.2 Evaluation Metrics

To compare against existing works, we adhere to the evaluation criteria from the original datasets for both our model and the baselines. For the UniGeo dataset, we utilise the top-10 accuracy metric, which measures the ratio of correct solution programs among the top ten predictions, aligning with the metric used by the authors of the UniGeo dataset. For the PGPS9K and Geometry3K datasets, we adopt a stricter metric, the top-3 accuracy, as recommended by the authors of the PGPS9K dataset. Note that our comparison involves matching the predicted solution program with the ground truth, which is more rigorous than merely comparing the numerical output derived from the solution program.[6]

6.4 Results and Discussions

6.4.1 Comparison with State-of-the-art Models (RQ-6.1)

We evaluate the performance of our GOLD model (*using T5-base as its problem-solving module*) against state-of-the-art (SOTA) methods in solving geometry math problems. The selected baselines for this comparison include: 1. **PGSPNet** [5]: it integrates a combination of CNN and GRU encoders, which generate an encoded vector of the diagram that serves as the input aligning with the logic form to the solver module. 2. **Inter-GPS** [60]: it parses both the problem text and the diagram into a formal language, subsequently feeding this into the solver. 3. **Geoformer** [1]: it utilises the

[6]This is grounded in the principle that a correct output can sometimes be produced by an incorrect solution program, indicating a failure in the model's understanding of the problem. For example, consider a problem where the correct answer is "5" and the correct program is "2 × 3 - 1". An incorrect program like "2 + 3" could still yield the correct output. Thus, generating the correct program is a more reliable indicator of the model's accurate problem comprehension.

Models	UniGeo CAL Test (%)	UniGeo Prv Test (%)	PGPS9K Test (%)	Geometry3K Test (%)
BERT2Prog	54.7†	48.0†	-	-
NGS	56.9†	53.2†	34.1‡	35.3‡
Geoformer	62.5†	56.4†	35.6‡	36.8‡
InterGPS	56.8	47.2	38.3	48.6
InterGPS (GT)	n/a	n/a	59.8‡	64.2‡
PGPSNet	53.2	42.3	58.8	59.5
PGPSNet (GT)	n/a	n/a	62.7‡	65.0‡
GAPS (Chapter 5)	68.5	97.2	37.2◊	38.0◊
GOLD	**75.2**	**98.5**	**60.6**	**62.7**
GOLD (GT)	n/a	n/a	65.8	69.1

Table 6.1: Comparison results on the test subsets of chosen datasets. PGPSNet reported models' performances using the ground truth diagram annotations, where these models have "(GT)" behind them. We re-implemented these methods to get performances without GT annotations. Note that UniGeo lacks GT diagram annotations, so relevant cells are "n/a". "†" and "‡" indicates the results are from [1] and [5], respectively. The performance of GAPS on the PGPS9K Test and Geometry 3K Test datasets, indicated by "◊" is evaluated using top-3 accuracy, the same metric employed by other models.

VL-T5 model for the purpose of diagram encoding, then servers encoded embeddings to the transformer. 4. **NGS** [11]: it uses the ResNet-101 for its encoding process, showcasing a different approach in handling the diagram encoding. 5. **Bert2Prog** [11]: it leverages BERT and ResNet as encoders and an LSTM network for generating. 6. **GAPS**: the model introduced in Chapter 5.

The results presented in Table 6.1 demonstrate that our GOLD model outperforms baselines across test subsets of all datasets. Specifically, when compared to Geoformer, our model exhibits a remarkable increase in accuracy: 12.7% on the UniGeo CAL and 42.1% on the UniGeo PRV. Compared to the SOTA model on PGPS9K and Geometry3K datasets, PGPSNet, the GOLD model surpasses it by 1.8% and 3.2% in accuracy, respectively. When using ground truth diagram annotations, the GOLD (GT) shows a improvement in accuracy on the PGPS9K and Geometry3K, with gains of 3.1% and 4.1% over PGPSNet (GT). Against InterGPS (GT), the improvements are at 6.0% and 4.9%, respectively. These results underline the effectiveness of the GOLD model in solving geometry maths problems.

When compared to the GAPS model as detailed in Chapter 5, the GOLD model consistently demonstrates better performance across all evaluated subsets.[7] Specifically,

[7]Please note that GAPS performances on PGPS9K test and Geometry3K test are reported in Top-3

the GOLD model exceeds the performance of the GAPS model by 6.7% on the UniGeo CAL subset and by 1.3% on the Prv test subset. The margins of improvement are even more pronounced in the PGPS9K and Geometry3K test subsets, underscoring the robustness of the GOLD model compared to the GAPS model.

Moreover, our GOLD model distinguishes itself from approaches like InterGPS and PGPSNet, which rely on logic-form representations to describe diagrams. In contrast, GOLD inputs natural language descriptions to LLMs to generate solution programs. Using natural language leads to improvements across all datasets compared to InterGPS and PGPSNet, as evidenced in Table 6.1. Furthermore, models like Geoformer and NGS primarily encode diagrams into vectors. These approaches fall short in providing precise descriptions of the diagrams and limit the adoption of LLMs, thus leading to worse performances compared to our GOLD model. This highlights the importance of detailed and accurate diagram representations for tackling geometry maths problems, where our GOLD model excels.

Worth mentioning is that the training for the symbol vector head, geometric primitive vector head, and relation-construction head of the GOLD model was exclusively conducted on the PGPS9K and Geometry datasets due to the lack of annotations in the UniGeo dataset. Despite this, the outstanding performance of the GOLD model on the test subset of UniGeo, as shown in Table 6.1, demonstrates its exceptional generalisation capability.

6.4.2 Ablation Study on Natural Language Description (RQ-6.2)

We assess our model's efficacy using three distinct diagram description formats: absence of diagram description, logic forms, and natural language descriptions. The comparative results are detailed in Table 6.2. When fine-tuning T5-base as the problem-solving module, Table 6.2 indicates that descriptions in natural language outperform those in logic-form, with 3.1% and 3.4% improvements on the test subsets of PGPS9K and Geometry3K, respectively.

Conversely, when using Llama2-13b-chat (Llama2) and CodeLlama-13b (CodeL-

accuracy, therefore the accuracy scores are worse than reported in Table 5.4.

	PGPS9K			Geometry3K		
	n/a	LF	NLD	n/a	LF	NLD
T5-base	22.3 ± *0.0*	57.5 ± *0.3*	**60.6** ± *0.3*	12.3 ± *0.0*	59.3 ± *0.5*	**62.7** ± *0.2*
Llama2-13b-chat	5.2 ± *0.0*	33.5 ± *0.4*	39.6 ± *0.2*	2.3 ± *0.0*	31.8 ± *0.3*	40.1 ± *0.4*
CodeLlama-13b	3.2 ± *0.0*	15.8 ± *0.0*	16.2 ± *0.0*	2.0 ± *0.0*	14.6 ± *0.0*	15.1 ± *0.0*

Table 6.2: Evaluation of the GOLD model on two datasets with no description (n/a), logic-forms (LF), and natural language descriptions (NLD). Both the mean and standard errors of the accuracy metrics are presented.

lama) as the problem-solving module, we implement instructions to guide the generation of answers. Since their generations differ from the ground truth, we opt to calculate the accuracy of choosing the correct option from given candidates. According to Table 6.2, using natural language descriptions enhances the accuracy of the Llama2 model compared to using logic forms, demonstrating the greater compatibility of our natural language descriptions with models like Llama2. However, neither natural language descriptions nor logic forms yield satisfactory outcomes with CodeLlama, possibly due to a mismatch between the training corpus of CodeLlama and the description formats.

Lastly, we conduct experiments by excluding relevant modules used to generate the natural language descriptions and solely inputting the problem text $\mathcal{T}$ into the problem-solving module. The results in Table 6.2 show a substantial decline in the performance of the GOLD model across all selected LLMs, highlighting the importance of diagram descriptions provided by relevant modules of the GOLD model in solving geometry maths problems.

6.4.3 GOLD's Performance on GeoEval Benchmark

Table 6.3 presents the top-10 accuracy achieved by the GOLD model on the GeoEval benchmark, introduced in Chapter 4. Note that we report the execution accuracy as no golden solution program provided by the GeoEval benchmark. We evaluate two variants of the GOLD model, as described in Section 6.4.2. The first variant, GOLD (T5), uses

	GeoEval Benchmark			
	GeoEval-2000 †	GeoEval-backward	GeoEval-aug †	GeoEval-hard
GOLD (T5)	62.3	6.8 ‡	40.1	0.0
GOLD (Llama2)	27.6	12.0 ‡	25.2	0.0

Table 6.3: The top-10 accuracy (%) of GOLD model achieved on GeoEval benchmark. The † denotes that GOLD was evaluated on partial subsets of GeoEval-2000 and GeoEval-aug, as it can only solve geometry math problems with available diagrams. Approximately 62% of problems in the GeoEval-2000 and GeoEval-aug subsets contain diagrams, sourced from the PGPS9K, UniGeo, GeoQA+, and Geometry3K datasets. The ‡ symbol indicates that GOLD utilizes only the problem text from the GeoEval-backward subset as input, since no corresponding geometric diagrams are available for that subset.

a fine-tuned T5-base as the problem-solving module, while the second variant, GOLD (Llama2), adapts the Llama2-13b-chat model directly as the problem-solving module. It is important to note that the GeoEval-backward subset does not have corresponding geometric diagrams, as these problems were created by reversing the original problems. To prevent the GOLD model from parsing the masked conditions directly from the original diagrams, we solely provide the problem texts as input when evaluating the model on the GeoEval-backward subset.

The results presented in Table 6.3 show that the GOLD (T5) model achieves a reasonable accuracy of 62.3% on the GeoEval-2000 subset. This performance can be attributed to the fact that a significant portion of the problems in the selected GeoEval-2000 subset are sourced from the UniGeo, PGPS9K, and Geometry3K datasets, which were used to train the GOLD (T5) model. Therefore, the model's familiarity with the problem types and linguistic patterns present in these datasets likely contributed to its proficiency on the GeoEval-2000 subset.

However, a notable observation is the substantial drop in GOLD's performance on the GeoEval-aug subset, where its accuracy decreases from 62.3% to 40.1%. This decline reveals a limitation in the model's ability to generalize effectively to problems with semantic diversity that deviates from its training corpus. Despite being rephrased versions of the original problems, the variations in linguistic expressions and problem formulations in the GeoEval-aug subset pose a challenge for the GOLD (T5) model, indicating its sensitivity to semantic divergences from its training data.

The results in Table 6.3 reveal that the GOLD (T5) model struggles with the GeoEval-backward subset, achieving a relatively low performance, with 6.8% accuracy. This limitation might stem from the model's lack of explicit training on backward reasoning tasks during the training stage. Backward reasoning, which involves working backwards from the given result to derive the masked conditions, is a crucial skill for solving geometric problems that require reversing the logical flow of problem-solving steps. Without dedicated training on this specific type of reasoning, the GOLD (T5) model may find it challenging to tackle problems in the GeoEval-backward subset effectively.

Furthermore, Table 6.3 shows that the GOLD (T5) model is unable to resolve problems in the GeoEval-hard subset. This limitation can be attributed to the model's inability to parse geometric diagrams involving solid geometry and analytic geometry concepts. These advanced geometric domains require specialized knowledge and problem-solving techniques that were not been represented in the model's training data. Consequently, the incorrect pre-parsing results obtained for these types of geometric diagrams ultimately harm the overall problem-solving accuracy of the GOLD (T5) model on the GeoEval-hard subset.

In addition, the results presented in Table 6.3 reveal that the GOLD (Llama2) model underperforms compared to the GOLD (T5) model on the GeoEval-2000 and GeoEval-aug subsets. This observation aligns with the conclusions drawn from Table 6.2, suggesting that the fine-tuning problem-solving module to generate solution program exhibits stronger performance than using instruction to guide LLMs to generate the reasoning process.

However, an interesting finding is that the GOLD (Llama2) model outperforms the GOLD (T5) model on the GeoEval-backward subset. This superior performance demonstrates that adopting the Llama 2 model as the problem-solving module, which leverages an instruction-based approach, can enhance the model's ability to resolve backward reasoning problems. In contrast, fine-tuning the T5 model appears to limit its performance on solving problems that require backward reasoning, potentially due to the model's training focus on forward reasoning tasks.

Furthermore, the results indicate that the performance gap (27.6% vs. 25.2%) between the GOLD (Llama2) model on the GeoEval-2000 and GeoEval-aug subsets is smaller than the gap (62.3% vs. 40.1%) observed for the GOLD (T5) model on these subsets. This finding suggests that the Llama2 model exhibits greater robustness to semantic diversity, as it is less affected by the linguistic variations present in the GeoEval-aug subset compared to the GOLD (T5) model. These observations highlight the potential benefits of leveraging instruction-based models like Llama2 for geometric problem-solving tasks. Such models may be better equipped to handle diverse reasoning requirements, including backward reasoning, while also demonstrating increased robustness to semantic variations in problem formulations.

The results also reveal that the GOLD (Llama2) model struggles with problems from the GeoEval-hard subset, similar to the GOLD (T5) model. This limitation can be attributed to the pre-parsing module of the GOLD framework, which is unable to accurately parse geometric diagrams involving solid geometry and analytic geometry concepts. Consequently, without correct descriptions of the geometric diagrams associated with these problems, it becomes extremely challenging for any problem-solving module, regardless of fine-tuned T5 or instruction-based Llama2, to accurately solve the problems. The pre-parsing module's failure to accurately interpret and represent the geometric concepts in these diagrams ultimately hinders the problem-solving module's ability to reason about the problem effectively.

6.4.4 Accuracy of the Extraction of geo2geo and sym2geo Relations (RQ-6.3)

Our analysis in Table 6.4 and measured by F1 metric, evaluates the accuracy of extracting geometric relations with and without emb$_{feat}$ and emb$_{spat}$ on PGPS9K test subset. We note that the pre-parsing stage achieves a high F1-score of 98.9%, ensuring accurate identification of symbols and geometric primitives for *sym2geo* and *geo2geo* relations extraction. However, when directly using $\mathbf{V}^{s,g}$ as vectors of symbols and geometric primitives (only using feature outputs from the pre-parsing step), the absence of emb_{feat} and emb_{spat} leads to a notable decrease in performance for both relations extraction. Conversely, the inclusion of either emb_{feat} and emb_{spat} results in improved performance. Table 6.4 further reveals that the extraction of both relation types reaches its highest F1-score when both embeddings are utilised. These results highlight the advantages of our approach in separately modelling symbols and geometric primitives, which proves to be more efficient in addressing the relation extraction of geometry maths problems.

emb_{feat}	emb_{spat}	pre-parsing	geo2geo	sym2geo
		98.9	65.2 ± *0.1*	58.6 ± *0.1*
✓		98.9	79.8 ± *0.3*	75.6 ± *0.5*
	✓	98.9	80.6 ± *0.4*	71.1 ± *0.2*
✓	✓	98.9	**93.7** ± *0.2*	**77.3** ± *0.1*

Table 6.4: The check mark (✓) indicates that the corresponding embedding is enabled. Note that "pre-parsing" is not influenced by emb_{feat} and emb_{spat}. Both the mean and standard errors of the accuracy metrics are presented.

6.4.5 Influence of feature_embedding and spatial_embedding on Geometry Problem Solving (RQ-6.4)

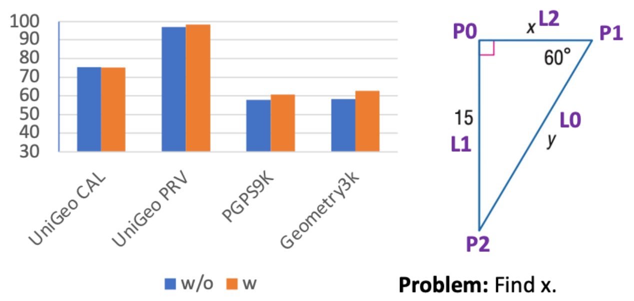
We conduct ablation study on *feature_embedding* and *spatial_embedding* in Table 6.5. To discard the use of (emb_{feat} and emb_{spat}), we directly use feature outputs from the pre-parsing step as vectors of symbols and geometric primitives, i.e., $\mathbf{V}^{s,g}$, to construct the *sym2geo* and *geo2geo* relations. We can observe that the GOLD model without any embedding performs the worst on all test subsets. However, when either one of embeddings (emb_{feat} or emb_{spat}) is added, the model's performance improves. Notably, the model equipped with both embeddings achieves the best performance.

6.4.6 Usefulness of the Additional geo2geo Relation (RQ-6.5)

Table 6.4 shows that the GOLD model accurately captures *geo2geo* relation, prompting us to investigate its impact on solving geometry math problems. The bar chart in Figure 6.2 indicates a notable decline in model performance on the PGPS9K and

emb$_{feat}$	emb$_{spat}$	CAL	PRV	PGPS9K	Geometry3K
		66.2 ± *0.3*	90.2 ± *0.2*	48.2 ± *0.5*	50.2 ± *0.3*
✓		71.5 ± *0.3*	93.2 ± *0.4*	55.0 ± *0.1*	58.1 ± *0.1*
	✓	72.8 ± *0.2*	93.0 ± *0.3*	56.3 ± *0.1*	58.0 ± *0.2*
✓	✓	**75.2** ± *0.3*	**98.5** ± *0.5*	**60.6** ± *0.3*	**62.7** ± *0.2*

Table 6.5: Program accuracy with or without *feature_embedding* and *spatial_embedding*. The check mark (✓) indicates that the corresponding embedding is enabled. T5-base is used as the problem-solving module for the GOLD model. Both the mean and standard errors of the accuracy metrics are presented.



Problem: Find x.

w/o: Line L2 and Line L0 cross at Point P1 has degree of 60. The length of Line L1 between Point P0 and Point P2 is 15. The length of Line L0 between Point P2 and Point P1 is y. The length of Line L2 between Point P0 and Point P1 is X. Line L2 is perpendicular with Line L1 at Point P0.

w: The diagram contains Line L0, which has endpoints: Point P1 and Point P2, the diagram contains Line L1, which has endpoints: Point P0 and Point P2, the diagram contains Line L2, which has endpoints: Point P0 and Point P1, ...

Figure 6.2: Top-left: the performance of the GOLD (using T5-base) with (**w**) and without (**w/o**) the *geo2geo*. Top-right: Geometry maths problem. Bottom: Predicted diagram description with and without the *geo2geo*. The same text between (**w**) and (**w/o**) is omitted for space consideration, where the red text is *geo2geo* relations.

Geometry3K datasets when *geo2geo* relations are omitted. However, this trend is less pronounced on the UniGeo datasets. This is likely because the PGPS9K and Geometry3K datasets often lack descriptions of geometric primitives in their problem texts. An example from the Geometry3K dataset, illustrated in Figure 6.2, demonstrates this issue: the problem text typically poses a question (e.g., "Find X") without extra information. Consequently, relying only on *sym2geo* relations leads to insufficient representation of essential diagram details.

6.5 Chapter Summary

This chapter addresses the challenge highlighted in Chapter 5, which emphasized the difficulties in accurately encoding and interpreting geometric diagrams when processing them alongside text. This chapter states that using natural language descriptions for geometric diagrams can provide accurate and comprehensible representations, bridging the gap between textual and visual information. Specifically, we introduced the GOLD model for automated geometry math problem-solving. GOLD uniquely converts geometric diagrams into natural language descriptions, facilitating direct integration with LLMs for problem-solving. A key feature of the GOLD model is its ability to separately handle symbols and geometric primitives, simplifying the establishment of relations between symbols and geometric primitives and among geometric primitives themselves.

To validate the statement, we assessed the effectiveness of GOLD relative to current state-of-the-art methods in Section 6.4.1. This comparison, conducted using the UniGeo CAL, PRV, PGPS9K, and Geometry3K datasets, demonstrated GOLD's superior performance, addressing Research Question **RQ-6.1**.

In Section 6.4.2, we explored research question **RQ-6.2** by comparing the performance of the GOLD model using three distinct diagram description formats: no diagram description, logic forms, and natural language descriptions. Results, displayed in Table 6.2, indicate that natural language descriptions outperform logic forms across three LLMs: T5, Llama2, and CodeLlama.

To address research questions **RQ-6.3** and **RQ-6.4**, we conducted ablation studies. As shown in Table 6.4 in Section 6.4.4, the highest F1-score for extracting relation types was achieved when both feature and spatial embeddings (emb_{feat} and emb_{spat}) were utilized. Furthermore, in Section 6.4.5, we demonstrated that the GOLD model using both embeddings performed best on all test subsets.

We further addressed research question **RQ-6.5** in Section 6.4.6. As illustrated in Figure 6.2, the absence of the *geo2geo* relation impacted GOLD's performance, confirming the critical role of the *geo2geo* relation in conveying essential details of geometric diagrams.

This work lays a foundation for future research on utilizing large language models to solve geometry math problems using natural language descriptions. Consequently, in Chapter 7, we investigate the impact of supplementing geometric diagrams with textual descriptions on the geometry math problem-solving performance of LLMs and MMs.

Chapter 7

Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving

In Chapter 6, we demonstrated that converting geometric diagrams into natural language descriptions seamlessly merges with the textual description of the problem. This conversion not only facilitates integration but also opens up the possibility of utilising large language models (LLMs) as generators for reasoning programs, which could enhance both interpretability and effectiveness. Such advancements prompt us to investigate the performance of the latest LLMs and Multi-Modal Models (MMs) in solving geometry problems. While recent progress in LLMs and MMs has shown their exceptional problem-solving abilities, their skill in addressing geometry problems—which requires the understanding of both textual and visual information—remains insufficiently explored. This chapter is mainly based on my accepted paper “GeoEval: Benchmark for Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving” in ACL 2024 Findings.

To address this gap, this Chapter evaluates 9 LLMs and MMs across these varied subsets reveals that the WizardMath model excels, achieving a 55.67% accuracy rate on the main subset but only a 6.00% accuracy on the challenging subset. This highlights the critical need for testing models against datasets on which they have not been pre-trained. Additionally, our findings indicate that GPT-series models perform more effectively on problems they have rephrased, suggesting a promising method for enhancing model capabilities.

This chapter is organised as follows: Section 7.2 conducts a detailed evaluation of the most recent LLMs and MMs using our GeoEval benchmark. The chapter concludes with Section 7.3, which offers a recap of the findings and contributions of this chapter.

7.1 Introduction

Rencently, large language models (LLMs) and multi-modal models (MMs) have demonstrated significant potential in handling complex reasoning tasks [226–228]. This potential has raised considerable interest in testing these advanced models across a variety of tasks, such as maths word problem solving [12] and physical problem solving [229]. Despite this interest, specific research on evaluating these models' effectiveness in geometry problem-solving remains scarce. Therefore, it is critical to develop a new, comprehensive benchmark that can effectively assess LLMs and MMs in geometry problem-solving, especially considering the potential exposure of existing public datasets during model training [230]. Comparing the performance of current LLMs and MMs on such a benchmark is essential, as it could yield valuable insights that further the development of models capable of tackling complex reasoning tasks.

This Chapter states that LLMs and MMs underperform on the automated geometry problem-solving task (See Statement (4) in Section 1.3). To validate this statement, we conduct extensive experiments using the GeoEval benchmark introduced in Chapter 4 to evaluate the proficiency of nine LLMs and MMs in solving geometry problems. This includes three LLMs: CodeGen2-16B [231], GPT-3.5 [219], and GPT-4 [76]; two LLMs specialized in mathematics: WizardMath-70B and WizardMath-7B-V1.1 [232]; and four MMs: llava-7B-V1.5 [233], Qwen-VL [234], mPLUG-Owl2 [235], and GPT-4V [76]. The findings reveal that GeoEval forms a challenging benchmark, with both LLMs and MMs struggling to resolve its complexities effectively.

This Chapter investigates the following research questions (RQ) to demonstrate the statement:

- **RQ-7.1**: How do the latest LLMs and MMs perform in geometry problem-solving tasks, and what does this indicate about their underlying mechanisms of understanding and reasoning?
- **RQ-7.2**: To what extent can training on specialised mathematical corpora improve the performance of models on geometry problem-solving, and what does this reveal about the role of domain-specific knowledge in AI reasoning?
- **RQ-7.3**: What is the impact of supplementing geometric diagrams with textual descriptions on the problem-solving performance of LLMs and MMs, and how do these descriptions affect the models' interpretability and accuracy?
- **RQ-7.4**: How do the latest LLMs and MMs integrate and apply external knowledge in solving complex geometry maths problems, and what methodologies enhance their effectiveness in such tasks?
- **RQ-7.5**: How do variations in the length of problem inputs affect the performance of the latest LLMs and MMs in geometry problem-solving, and what does this suggest about the models' processing and comprehension capabilities?
- **RQ-7.6**: What is the relationship between the complexity of geometry problems and the performance of the latest LLMs and MMs, and how can models be optimised to handle increasingly complex problems?

7.1.1 Contributions

To summarise, the contributions of this chapter are:

1. We conduct comprehensive evaluation of the latest LLMs and MMs, we believe the evaluation results analysis provide the first comprehensive quantitative assessment of the latest LLMs and MMs in the domain of geometry problem-solving.

7.2 Evaluating LLMs and MMs on GeoEval benchmark

7.2.1 Experimental Setup

In this chapter, we deliberately select state-of-the-art LLMs and MMs that are widely recognised for their advanced capabilities, including

- **LLMs Specialised in Programming Code**: We include CodeGen2-16B model [231], which is renowned for its proficiency in understanding and generating programming code, offering insights into its adaptability to solve geometry maths problems.

- **LLMs with a Focus on Mathematics**: This includes WizardMath-7B-V1.1 and WizardMath-70B [232], explicitly pre-trained on mathematical corpora. Their inclusion allows for an assessment of models that have been fine-tuned to tackle complex mathematical problems.

- **LLMs Designed for a Broad Range of Topics**: Models such as GPT-3.5 [219] and GPT-4 [76] exemplify the advanced commercial LLMs engineered to encompass a broad range of topics.

- **Multi-Modal Models (MMs) with Diverse Decoders**: Given the ubiquity of ViT architecture [161] as the vision encoder in MMs, we select models that integrate ViT with various LLMs as decoders. This includes llava-7B-V1.5 [233] with Vicuna [236], Qwen-VL [234] using Qwen [237], mPLUG-Owl2 [235] with LLaMA [75], and GPT-4V [76].

These models are evaluated through a zero-shot approach, utilising straightforward instruction prompts to directly assess their geometry problem-solving capabilities without further fine-tuning specifically for our benchmark.[1]

[1]Details on the hyperparameters used for these models and the prompt design are available in Appendix E.1 and Appendix E.2.

7.2.2 Evaluation Metric

Building upon the approach by MathVista [12], we first input the generated sequence from the model into GPT-4 to extract the target value or option letter. To enhance the precision of our answer extraction, we formulate intricate rules for post-processing the outcomes in cases where GPT-4 falls short. This approach has enabled us to attain an extraction accuracy surpassing 97%, similar to the success rate reported in MathVista [12]. Details on the crafted prompts and the extraction guidelines are available in Appendix E.3.

The extracted results are compared against the golden answers to determine the final performance metric. Given the model's intention to produce responses in varying formats, either as the precise answer (for instance, "3.15") or as the corresponding option letter (such as "A"), we regard a prediction as accurate if it either matches the golden answer or the golden option letter.

7.2.3 Experimental Results (RQ-7.1)

	GeoEval-2000		GeoEval-backward	GeoEval-aug	GeoEval-hard
Model	A (%)	T (%)	A (%)	A (%)	A (%)
CodeGen2-16B ◊	28.76	22.06	5.10	8.50	5.66
GPT-3.5 ◊	24.71	21.27	22.66	41.25	22.33
GPT-4 ◊	27.95	43.86	26.00	45.75	10.10
WizardMath-70B ◊	**55.67**	34.20	28.66	37.75	6.00
WizardMath-7B-V1.1 ◊	54.78	32.76	32.66	**47.75**	6.00
llava-7B-V1.5	12.80	21.01	11.33	20.25	20.30
Qwen-VL	25.60	25.97	5.66	22.25	21.66
mPLUG-Owl2	37.76	n/a	**35.33**	38.00	**22.66**
GPT-4V	37.22	**43.86** ‡	26.00	45.75	10.10

Table 7.1: Accuracy scores of models on our GeoEval benchmark. The "◊" refers to all LLMs. The "A" signifies the overall accuracy across all problems, while "T" denotes the accuracy for problems containing only texts without diagrams. "n/a" indicates that scores are unavailable due to models cannot process text-only inputs. The "‡" notes that the accuracy figures for GPT-4V are derived from GPT-4, as GPT-4V does not support image-free inputs.

In this section, we present the accuracy achieved by models on our GeoEval benchmark. Table 7.1 highlights that models pre-trained on a math-specific corpus tend to outperform others. Furthermore, except for llava-7B-V1.5 and Qwen-VL, multi-modal models (MMs) generally exceed the performance of large language models (LLMs).

7.2.3.1 Comparison among LLMs

When reviewing the performances of LLMs as detailed in Table 7.1, it becomes evident that models pre-trained on mathematical corpora demonstrate superior efficacy in solving geometry maths problems compared to those trained on general corpora. Specifically, evaluating on all problems of GeoEval-2000 subset (marked as "A" in the table), WizardMath-70B leads with an accuracy of 55.67%, while WizardMath-7B-V1.1 closely follows with a 54.78% accuracy, outperforming other LLMs. Conversely, GPT-4, GPT-3.5, and CodeGen2-16B report notably lower accuracies, all under 30.00%. Focusing on questions solely based on problem text within the GeoEval-2000 subset (indicated as "T" in the table), GPT-4 emerges as the frontrunner, securing the highest accuracy of 43.86%, with WizardMath models also surpassing the 32.00% accuracy. These findings underscore the enhanced proficiency of models pre-trained on math-specific corpora in tackling geometry maths problems, particularly when problems are well-described textually, as evidenced by GPT-4's leading performance.

In the GeoEval-backward subset, WizardMath-7B-V1.1 excels with the highest accuracy of 32.66%, closely followed by WizardMath-70B at 28.66%. This drop in performance across all LLMs, compared to the GeoEval-2000 results, highlights a collective weakness in backward reasoning capabilities. For the GeoEval-aug subset, WizardMath-7B-V1.1 again tops the leaderboard with an accuracy of 47.75%, with GPT-4 not far behind at 45.75% accuracy. Lastly, within the GeoEval-hard subset, all models, excluding GPT-3.5, exhibit relatively low accuracies, indicating a broad difficulty in addressing the most challenging solid geometry and analytic geometry problems.

7.2.3.2 Comparison among Multi-Modal Models

Table 7.1 shows that among the MMs, GPT-4V and mPLUG-Owl2 consistently outperform their counterparts across all subsets. Specifically, within the GeoEval-2000 subset, mPLUG-Owl2 leads with an accuracy of 37.76%, closely followed by GPT-4V at 37.22%, with the remaining MMs fall behind at lower accuracies. Specifically, Qwen-VL and llava-7B-V1.5 achieve accuracies of 25.60% and 12.80%, respectively. When examining problems that only involve texts, GPT-4V achieves a 43.86% accuracy, surpassing llava-7B-V1.5 (21.01%) and Qwen-VL (25.97%).

In the GeoEval-backward subset, mPLUG-Owl2 tops with the accuracy of 35.33%, with GPT-4V following at 26.00% accuracy. This performance shows a notable lack in backward reasoning skills, as illustrated by the diminished results of llava-7B-V1.5 and Qwen-VL in this category. Moving to the GeoEval-aug subset, GPT-4V leads with an impressive 45.75% accuracy, with mPLUG-Owl2 at the second place with 38.00% accuracy. Both Qwen-VL and llava-7B-V1.5 show comparable performances in this subset. Lastly, within the GeoEval-hard subset, mPLUG-Owl2 demonstrates the highest efficacy with a 22.66% accuracy, closely followed by Qwen-VL and llava-7B-V1.5. Surprisingly, GPT-4V records a lower accuracy of just 10.10%, highlighting the challenging nature of GeoEval-hard subset and the varied capabilities of MMs in addressing the most difficult problems.

7.2.3.3 Comparison between LLMs and Multi-Modal Models

In the GeoEval-2000 subset, specifically for problems that only include texts, GPT-4's performance exceeds the top MMs, Qwen-VL, by 17.89%. This is attributed to the MMs' inability to access geometric diagrams, which likely hinders their comprehension of the problems. Moreover, when evaluating across all problems of the GeoEval-2000 subset, WizardMath-70B surpasses the best MMs, Qwen-VL, by 17.91% in accuracy. However, MMs like GPT-4V and mPLUG-Owl2 achieve higher accuracy than LLMs not pre-trained on mathematical content. This underscores the value of mathematical pre-training for excelling in geometry problem-solving. Notably, GPT-4V's accuracy on all GeoEval-2000 problems is 9.27% higher than GPT-4's, suggesting GPT-4V's superior capability in solving geometry problems with diagrams.

This pattern persists in the GeoEval-aug subset, where WizardMath-7B-V1.1, a model trained on a mathematical corpus, achieves the highest accuracy at 47.75%. Conversely, mPLUG-Owl2 leads in the GeoEval-backward and GeoEval-hard subsets, with accuracies of 35.33% and 22.66%, respectively. Given that GeoEval-aug rephrases questions from GeoEval-2000, it implies both subsets might have been exposed to the models during their pre-training phase. In contrast, GeoEval-backward and GeoEval-hard subsets are less likely to have been previously exposed. This suggests that WizardMath-7B-V1.1 excels with familiar geometry math problems, while mPLUG-Owl2 demonstrates a robust capability in tackling unseen geometry problems. This is further evidenced by the low performance of WizardMath models on the GeoEval-hard subset, where both models only achieve an accuracy of 6.00%.[2]

7.2.3.4 Analysis on the Best Model (RQ-7.2)

Table 7.1 shows that GPT-4, the leading LLMs, records the highest accuracy on the GeoEval-aug subsets, though it only secures a 27.95% accuracy on the GeoEval-2000 subset. A similar pattern of improvement is noted for the GPT-3.5 model, which sees its accuracy jump from 24.71% on the GeoEval-2000 subset to 41.25% on the GeoEval-aug subset. This improvement aligns with the involvement of GPT-3.5 in generating the GeoEval-aug subset, suggesting that the capabilities of GPT-3.5 and GPT-4 in addressing geometry math problems benefit from their use in rephrasing geometry question texts.

While WizardMath-70B and WizardMath-7B-V1.1, both pre-trained on a mathematical corpus, demonstrate superior performance on the GeoEval-2000 subset, they show a marked decline in accuracy across the other subsets, with the most decreases observed on the GeoEval-hard subset. This indicates that although pre-training on a mathematical corpus is crucial for solving geometry math problems, it may not be enough.

In contrast to the variances in accuracy observed among LLMs across different

[2]Please see Appendix E.4 for detailed accuracy scores for models across various academic subjects.

subsets, the top-performing multi-modal model, mPLUG-Owl2, maintains relatively stable accuracies with scores of 37.76% on the GeoEval-2000, 35.33% on the GeoEval-backward, and 38.00% on the GeoEval-aug subsets. Additionally, the performance of GPT-4V on the GeoEval-aug subset surpasses its accuracy on the GeoEval-2000 subset, mirroring the trends observed with GPT-4 and GPT-3.5, further illustrating the enhanced effectiveness of GPT-series models when engaged in rephrasing the content of geometry questions.

7.2.4 Benefit from the Geometric Diagram Descriptions (RQ-7.3)

Models	✗	✓
GPT-4V	40.28	45.61 (*+5.33*)
WizardMath-7B	38.10	56.83 (*+18.73*)

Table 7.2: Comparison of models with (✓) and without (✗) geometric diagram descriptions.

To assess the impact of including geometric diagram descriptions on models' ability to comprehend geometric diagrams and solve related problems, we selected a sample of 300 questions with geometric diagram descriptions from the GeoEval-2000 subset. We then evaluated the performance of two models, GPT-4V and WizardMath-7B-V1.1, on these questions, both with and without the use of geometric diagram descriptions. The results in Table 7.2 indicate that GPT-4V's accuracy decreases by 5.33% without the diagram descriptions. More significantly, WizardMath-7B's accuracy falls by 18.73% in the absence of these descriptions. This evidence suggests that supplemental geometric diagram descriptions enhance models' efficiency in solving geometry math problems, particularly benefiting LLMs.

7.2.5 External Knowledge Required (RQ-7.4)

In the GeoEval benchmark, certain questions require external knowledge, such as the value of π, which is not typically included in the problem text. This necessitates models to have pre-existing knowledge to accurately solve these problems. Figure 7.1 assesses

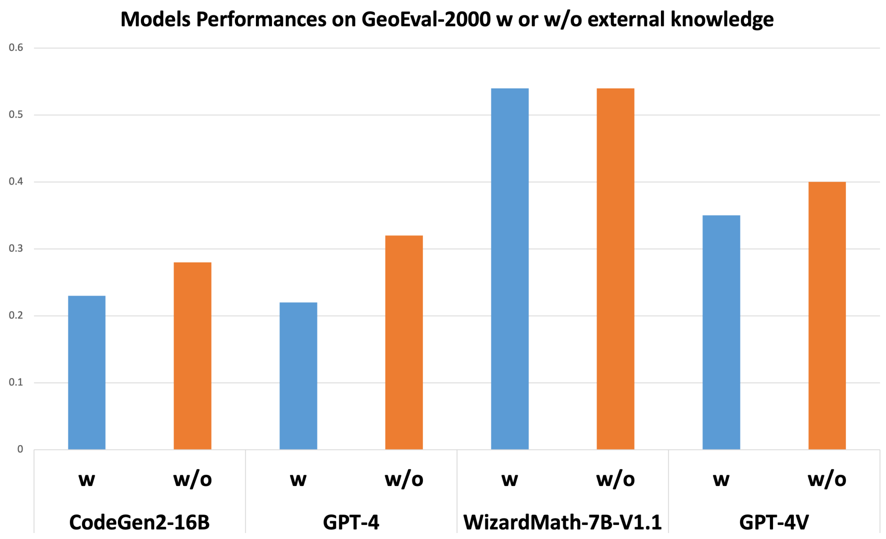


Figure 7.1: Comparison of models requiring external knowledge ("w" in blue colour) and those do not ("w/o" in orange colour).

the performance of four models on problems differentiated by the need for external knowledge, identified through a heuristic approach that classifies problems according to whether its solutions requires constants.

Figure 7.1 shows that the WizardMath-7B-V1.1 model maintains consistent accuracy on GeoEval-2000 subset, regardless of the requirement for external knowledge, unlike other models, which perform better on problems without such requirements. This consistency in WizardMath-7B-V1.1's performance is likely due to its pre-training on a maths-specific corpus, providing it with the necessary knowledge to resolve geometry maths problems effectively. In contrast, models trained on general corpora may not possess this specialised mathematical knowledge, hindering them from solving the problems correctly.

7.2.6 Performances According to Different Problem Lengths and Varied Complexities (RQ-7.5, RQ-7.6)

Figure 7.2 shows how models perform with inputs of different lengths. Performance slightly varies for problems ranging from 80 to 100 characters, but there's a clear trend of decreasing accuracy as problem length increases. This is expected, as longer questions

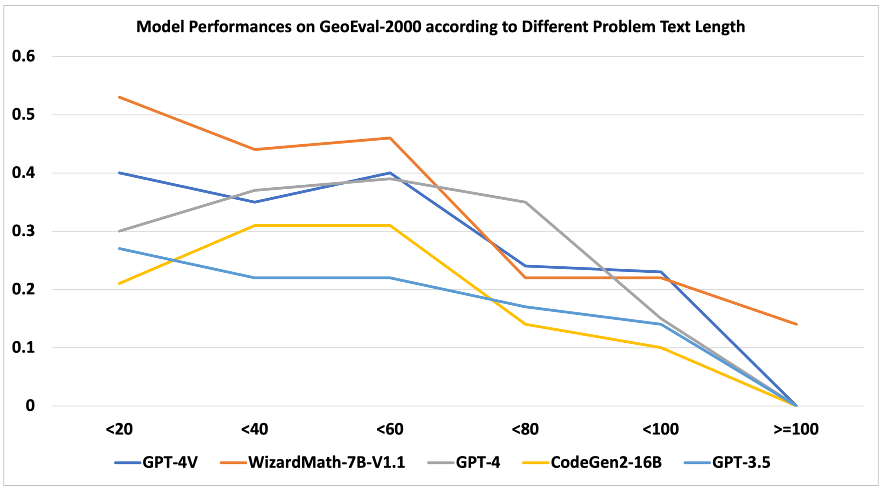


Figure 7.2: Models performances on GeoEval-2000 subset according to different question lengths.

typically involve more complex geometry maths problems, challenging the models more as the length grows. The figure also points out that the WizardMath-7B-V1.1 model is notably more adept at handling longer questions, with GPT-4V and GPT-4 showing relatively stable accuracy for increased question lengths. On the other hand, GPT-3.5 and CodeGen2-16B perform less effectively on lengthy questions.

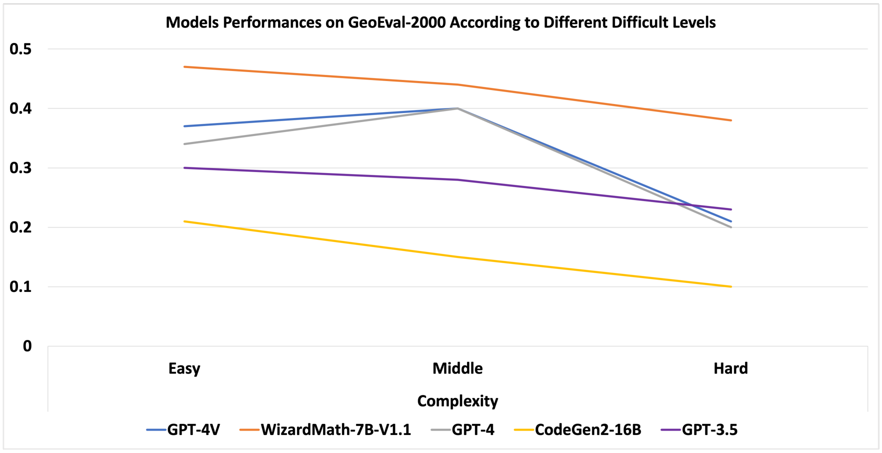


Figure 7.3: Model performances on GeoEval-2000 subset according to different complexity levels.

Upon the analysis in Figure 7.3, similar to the observations made in Figure 7.2 regarding input lengths, we delve into the models' performances as they relate to the complexity of geometry maths problems. Figure 7.3 presents the performance of models across varying levels of problem complexity. It is evident that as the complexity of geometry problems escalates, the accuracy of the models correspondingly diminishes.

7.3 Chapter Summary

In this chapter, we evaluated nine cutting-edge LLMs and MMs using the GeoEval benchmark to validate the assertion that these models underperform in solving geometry problems. Utilizing the proposed GeoEval benchmark from Chapter 4, we assessed the proficiency of these state-of-the-art models in tackling geometry math problems. Our analysis, presented in Table 7.1, reveals that both LLMs and MMs continue to struggle in this domain. Notably, even the top-performing model, WizardMath-70B, achieved only 55.67% accuracy on the GeoEval-2000 subset, addressing Research Question **RQ-7.1**.

In Section 7.2.3.4, we address Research Question **RQ-7.2** by analyzing the highest-performing model on the GeoEval benchmark, WizardMath-70B. Our investigation concludes that models pre-trained on mathematical corpora exhibit superior performance in automated geometry problem-solving tasks due to their implicit integration of necessary mathematical knowledge. However, despite the critical importance of such pre-training, it may not be sufficient on its own.

Section 7.2.4 emphasizes the importance of integrating descriptions of geometric diagrams, which can enhance LLMs' and MMs' ability to comprehend and solve geometry problems. This finding is supported by the results of the ablation study presented in Table 7.2, directly addressing Research Question **RQ-7.3**.

To address Research Question **RQ-6.4**, we conducted a comparative analysis of four representative models to assess their performance on geometry problems requiring external knowledge versus those that do not. As shown in Figure 7.1, we observed a performance decline in CodeGen2-16B, GPT-4, and GPT-4V on problems requiring external knowledge, except for the WizardMath-7B-V1.1 model. WizardMath models excel in automated geometry problem-solving due to their inherent integration of necessary mathematical knowledge, directly addressing Research Question **RQ-7.4**.

In Section 7.2.5, we addressed both Research Questions **RQ-7.5** and **RQ-7.6** by examining model performance concerning varying question lengths and complexity levels. As illustrated in Figure 7.2 and Figure 7.3, it is evident that model performance diminishes as question length and complexity increase.

Part IV

Conclusion and Future Work

Chapter 8

Conclusion and Future Work

This thesis concentrates on investigating and advancing the application of deep learning methods in solving complex mathematical reasoning problems, with a particular focus on text-based automated numerical reasoning task and automated geometry maths problem-solving task.

The traditional approaches for text-based numerical reasoning task often involve using solution programs to represent the reasoning process. However, a common limitation in these models is their inability to distinguish between operators and operands within solution programs, hindering their adaptability to more complex tasks. Addressing this gap, we introduce the ELASTIC model in Chapter 3. ELASTIC uniquely separates the generation of operators and operands, thereby reducing cascading errors in intricate reasoning processes. Its design also allows flexibility in the number of operands per operator, making it suitable for various domains. Our experiments demonstrate that ELASTIC outperforms existing baselines in this field.

While text-based automated numerical reasoning has progressed significantly, the field of mathematical reasoning on multi-modal data, especially the geometry maths problem-solving, is relatively at early stage. Recognising the need for advancement in this area, we have created the GeoEval benchmark in Chapter 4, which is developed to assess the geometry problem-solving capabilities. Specifically, GeoEval comprises four distinct subsets, each designed to facilitate a thorough evaluation. This benchmark facilitates a deeper investigation into the performance of models on solving geometry maths problems. In addition, we have evaluated proposed GPAS model in Chapter 5 and GOLD model in Chapter 6 on GeoEval benchmark, showcasing it as a challenging benchmark.

Prior research in solving geometry math problem often focused on a single type of geometry problem due to the complexity introduced by diverse domain-specific languages. However, a substantial overlap in mathematical knowledge across different problem types exists [1]. To overcome this challenge, we have developed methods to automatically solve geometry maths problems, as detailed in Chapter 5. Our proposed model, GAPS, is a universal solver designed to tackle various types of geometry problems. It functions based on the fact that solution programs follow a unified pattern, comprising operators (like arithmetic operators and geometric theorems) and operands (such as numbers and geometric elements). GAPS includes a problem-type classifier that identifies different geometry problem types, enabling the model to select appropriate tokens from various domain-specific languages. This approach allows GAPS to generate operators and operands tailored to each problem type. Experimental evaluations across multiple geometry maths datasets have shown that GAPS achieves state-of-the-art results. Furthermore, its effectiveness is enhanced through training on augmented datasets, leveraging its ability to solve diverse geometry problems with a single solution program generator, supported by the problem-type classifier.

However, unlike other image-text as input, geometric diagrams present unique challenges due to their complex interrelations among components, which cannot be accurately captured by simply converting diagrams into vectors. This limitation hinders the effectiveness of models in solving geometry maths problems. Prior attempts to address this involved parsing diagrams into formal language for greater precision and interpretability. However, these methods suffer from limited scalability and complexity in rule design. Additionally, they necessitate specific solvers compatible with formal languages, rendering them incompatible with current large language models (LLMs). To overcome these challenges, we introduce the GOLD model in Chapter 6. GOLD converts geometric diagrams into natural language descriptions. This approach not only enhances interpretability but also bridges the gap between image and text modalities, facilitating integration with LLMs. Consequently, the GOLD model leverages the advanced capabilities of LLMs for problem-solving. Our experimental findings demonstrate that the GOLD model outperforms existing state-of-the-art models in this domain.

Recently, with the advance of large language models (LLMs) and multi-modal models (MMs), to conclude the research in this thesis, in Chapter 7, we evaluate 9 cutting-edge LLMs and MMs using the GeoEval benchmark, we underscore the critical role of mathematical corpus pre-training for effective geometry problem resolution. Additionally, our analysis reveals that GPT-series models exhibit improved performance on geometry problems they have rephrased, pointing to the potential benefits of self-rephrasing in problem-solving.

The outline of this chapter is as follow. We summarise the main contributions of this thesis in Section 8.1. Next we discuss the main findings and conclusions of this thesis in Section 8.2. Finally, we discuss the ongoing future work in Section 8.3.

8.1 Contributions

The main contributions of this thesis are as follows:

- In Chapter 3, our investigation revealed that current methods struggle with complex mathematical reasoning tasks, primarily due to limited extensibility and a lack of consideration for the unique nature of solution programs in these tasks. Addressing this, we introduced the num**E**rica**L** re**AS**oning with adap**T**ive symbol**I**c **C**ompiler (ELASTIC) model. ELASTIC separates the generation of operators and operands, thereby reducing errors in complex reasoning processes. Its innovative design includes distinct modules and a Memory Register, enhancing stability in challenging numerical reasoning problems. Additionally, ELASTIC is a domain-agnostic approach, supporting a wide range of operators, ensures its flexible extensibility. Our experiments confirm that ELASTIC outperforms the existing state-of-the-art baselines.

- In Chapter 4, we introduced the GeoEval benchmark, a comprehensive collection specifically designed for automated geometry math problem-solving task. GeoEval features its comprehensive variety, sourced from seven public datasets and formatted uniformly to encompass a wide range of geometric shapes.

- In Chapter 5, our focus expanded from the concepts introduced in Chapter 3, extending our methodology to address mathematical reasoning tasks in geometry maths problems, which involve both textual and geometric diagram as the input. We introduced the GAPS model, featuring an innovative encoder that adeptly processes both text and diagrams. A key advancement of GAPS is its unified framework, which efficiently handles a diverse array of geometry maths problems without the need for distinct program generators for different problem types. Our experimental findings highlight GAPS' superior performance in solving geometry maths problems, outperforming other methods in this field.

- In Chapter 6, we delved into optimising the combination of problem texts and geometric diagrams for solving geometry maths problems. Our solution, the GOLD model converts geometric diagrams into natural language descriptions. This approach gets rid of the need for complex rule creation while facilitating smooth integration with large language models (LLMs). Our experimental findings not only demonstrate GOLD's substantial superiority over preceding methods but also highlight the effectiveness of converting images into text for improved modality fusion.

- In Chapter 7, we conduct extensive experiments using the GeoEval benchmark to evaluate the proficiency of 10 LLMs and MMs in solving geometry problems. This includes three LLMs: CodeGen2-16B [231], GPT-3.5 [219], and GPT-4 [76]; two LLMs specialised in mathematics: WizardMath-70B and WizardMath-7B-V1.1 [232]; and five MMs: llava-7B-V1.5 [233], Qwen-VL [234], mPLUG-Owl2 [235], and GPT-4V [76]. The findings reveal that GeoEval forms a challenge benchmark, with both LLMs and MMs struggling to resolve its complexities effectively.

8.2 Findings and Conclusions

The main finding and conclusion of this thesis are that *the enhancement of text-based numerical reasoning and geometry math problem-solving capabilities can be achieved through the innovative architecture of the solution program decoder and the straightforward representational approaches of the diagrams.* To ensure success in these tasks, the following considerations are crucial in method design:

- (Chapter 3): In the context of textual mathematical reasoning, our approach of segregating the generation of operators and operands effectively reduces errors in complex problems, thereby enhancing the model's robustness. This strategy is also domain-agnostic, accommodating a wide range of operators and thus boosting its generalisation capability. Our method demonstrates superior performance, achieving execution accuracy of 68.96% on the FinQA dataset and 65.21% on the MathQA dataset for program accuracy, surpassing previous state-of-the-art models.

- (Chapter 3): Preserving and updating intermediate results is crucial for generating lengthy solution programs. By employing the Maximum Memory Departing Distance (M-MDD) metric, we demonstrate that ELASTIC with a Memory Register outperforms its counterpart without this feature when M-MDD exceeds 5.

- (Chapter 5): Effectively addressing various types of geometry maths problems simultaneously is important, as evidenced by our GAPS model. It incorporates a problem-type classifier, allowing a single solution program generator to adaptively handle different problem types.

- (Chapter 5): Training models on a wide range of geometry maths problems boosts their performance and generalisation capabilities across diverse problem types. This approach has proven to be more effective than state-of-the-art methods, as validated by our experimental results.

- (Chapter 6): Accurate representation of both components and relationships in geometric diagrams is essential for successful problem solving. Our research shows that converting geometric diagrams into natural language descriptions not only enhances interpretability but also effectively bridges the gap between different modalities, namely geometric diagrams and problem texts.

- (Chapter 6): Beyond the relationships among geometric primitives, including details about the primitives themselves, like their types and spatial information, can enhance the model's performance in automatically solving geometry maths problems.
- (Chapter 7): Despite the advancements in LLMs and MMs have demonstrated potential in handling complex reasoning tasks, they still fall short in solving geometry maths problems. Our GeoEval benchmark (Chapter 4) suggests that both LLMs and MMs struggles to resolve its complexities effectively.

In the following part of this section, we will elaborate on these findings and provide a detailed explanation.

8.2.1 Limitations of State-of-The-Art Methods on Text-based Automated Numerical Reasoning Task

In Chapter 3, we identified two major challenges with existing methods in handling complex text-based numerical reasoning task. First, these methods struggle with long reasoning programs characterised by diverse operators and dynamic operand numbers, leading to cascading errors due to their inability to separate operator and operand generation. Second, they exhibit limited extensibility regarding operators, attributed to either model architecture limitations or the program representation format, which hinders their application in different domains. Consequently, we proposed a novel network architecture that segregates operator and operand generation in reasoning programs, stating that this approach minimises cascading errors and enhances adaptability by integrating a broader range of operators (see **Statement (1)** in Section 1.3). Based on this principle, we proposed the ELASTIC model, to fully evaluate our model, we answered the following research questions proposed in Section 3.1. To address these research questions, we took extensive experiments to compare our ELASTIC model with leading methods using two datasets specifically designed for assessing automated mathematical reasoning abilities (refer to Section 3.3 for details).

We initiated our comparison by evaluating ELASTIC against previous methods on two benchmarks, FinQA [10] and MathQA [9]. This effectively addresses research question **RQ-3.1**, demonstrating that ELASTIC outperforms other methods, achieving the highest scores on both datasets (refer to Table 3.2 for detailed results).

Next, we conducted a more detailed comparison with a specific model. As discussed in Section 3.4.1, FinQANet [10], which utilises the same encoder as ELASTIC, showed inferior performance. This underscores the efficacy of our method's separation design, addressing research question **RQ-3.2**. Additionally, we focused on operator generation accuracy, disregarding operand correctness. The findings, illustrated in Figure 3.2, reveal that operator accuracy consistently surpasses overall program accuracy across different program steps. This highlights the benefits of segregating the generation processes can disentangle the errors from operators and operands.

Furthermore, we addressed **RQ-3.3** in Section 3.4.1 by comparing ELASTIC with other methods across two distinct datasets. The FinQA dataset focuses on the financial domain, while MathQA focuses on elementary maths word problems. Our results, detailed in Table 3.2, demonstrate that ELASTIC surpasses other methods, showcasing its robust performance even with datasets from different domains.

Following our comparison with state-of-the-art methods, we examined the robustness of ELASTIC, addressing research question **RQ-3.4**. In Section 3.4.2, we explored the impact of solution program length on model performance. Our findings clearly show that ELASTIC consistently outperforms the previous best method, FinQANet, across various program lengths. This indicates that ELASTIC is more resistant to cascading errors in longer solution programs.

To further examine the robustness of ELASTIC, which aligns with research question **RQ-3.4**, we conducted an evaluation using a challenging dataset we curated. This dataset features solution programs with operators having varying numbers of operands. The ELASTIC model, while performing slightly lower on this challenging dataset compared to the original one, still demonstrated strong results. This led us to conclude that the ELASTIC model excels in handling problems necessitating the generation of operators with diverse operand counts.

To improve ELASTIC's performance with lengthy solution programs, we incorporated a novel feature: the memory register (MR), detailed in Section 3.2.3.5. To assess its impact, we addressed research question **RQ-3.5** through an ablation study on the MR. The results, as shown in Table 3.4, indicate that ELASTIC with MR outperforms its counterpart without MR. Additionally, we introduced the Memory Departing Distance (MDD) and maximum Memory Departing Distance (M-MDD) in Section 3.4.3 to evaluate ELASTIC's efficacy in utilising results from earlier sub-program steps. The findings demonstrate that ELASTIC with MR yields better outcomes when referencing results from more distant steps than ELASTIC without MR.

8.2.2 Generalise the Maths Reasoning Ability From Text-based Data to Image-text Data

In Chapter 5, our investigation focused on adapting our proposed methods to solve maths problems with an additional component – geometric diagrams. Recent research has largely concentrated on a single type of geometry maths problem, often due to the need for different domain-specific languages for other types. Additionally, while some state-of-the-art methods have tried to extend models successful in textual mathematical reasoning to geometry problems, few have achieved satisfactory results. This led us to state that we could effectively leverage the unique features of various geometry problem types without encountering decline (refer to **Statement (2)** in Section 1.3). In line with this hypothesis, we introduced the Geometry-Aware Problem Solver (GAPS), a model tailored to autonomously tackle different types of geometry maths problems simultaneously. We aimed to validate this hypothesis by addressing specific research questions proposed in Section 5.1.

In Section 5.5.1, we resolved research question **RQ-5.1** by comparing our GAPS model against leading methods in two types of geometry problems: calculation and proving. As demonstrated in Table 5.2, GAPS notably outperformed the other baseline models.

In the following Section 5.5.2, we compared the accuracy of our GAPS model with other baseline models based on sub-program types. Figure 5.4 clearly shows that GAPS excelled over the baselines in both main problem types and their respective sub-tasks, addressing research question **RQ-5.2**.

In our analysis, we noted a notable disparity in GAPS's performance on proving versus calculation problems, aligning with research question **RQ-5.3**. Section 5.5.3 reveals that most errors in calculation problems stem from incorrect operators, while in proving problems, errors are more often due to incorrect operands. This suggests that calculation problems present a more diverse and challenging set of solution programs. Consequently, GAPS performs better on proving problems than on calculation problems, thereby addressing research question **RQ-5.3**.

We further investigated our hypothesis regarding the impact of training on diverse geometry maths problems with different domain-specific languages. In Section 5.5.4, we incorporated PGPS9K [5] as an additional training dataset, comprising 6,131 newly created problems and 2,891 problems from the Geometry3K dataset [60]. We assessed the model's performance using three datasets: solely UniGeo, solely PGPS9K, and a combination of both. The results, shown in Table 5.4, indicate improvements in the GAPS model's performance when trained on the combined dataset. Thus, we confirm **RQ-5.4**: incorporating problems with various types indeed boosts performance, rather than hindering it.

In Section 5.5.5, we explored the importance of the problem-type classifier in GAPS, which allows simultaneous generation of solution programs for various types of geometry maths problems, thereby addressing research questions **RQ-5.5**, **RQ-5.6**, and **RQ-5.7**. Our ablation study, as shown in Table 5.5, revealed that including the problem-type classifier enhances GAPS's performance on both the UniGeo dataset [1] and a combination of UniGeo and PGPS9K datasets, while its absence leads to a marked decrease in accuracy. Furthermore, optimising the number of problem types resulted in maximised benefits for GAPS. Additionally, training loss convergence analysis indicated that GAPS equipped with the problem-type classifier achieves faster convergence compared to the version without it, confirming the effectiveness of this feature in addressing research questions **RQ-5.5**, **RQ-5.6**, **RQ-5.7**.

We further addressed research question **RQ-5.8** by analysing the effectiveness of updating the cache token with the query vector used for generating the final operand in the sub-program, as detailed in Table 5.7. This approach yielded the highest accuracy, validating our method. Additionally, due to our unique design that separates operator and operand generation, we developed a specialised algorithm for hierarchical beam search, as described in Section 5.3.4.4. The results in Table 5.8 demonstrate that employing beam search enhances GAPS's accuracy, effectively answering research question **RQ-5.9**.

To address research question **RQ-5.10**, as detailed in Section 5.5.9, we performed an ablation study by removing the geometry elements enhancement feature and evaluated its impact on the UniGeo dataset. The results, shown in Table 5.9, highlight the advantages of incorporating the geometry elements enhancement method. Further, we presented a case study that visualises the probability distributions during operand generation with and without this enhancement, as depicted in Figure C.2. The figure indicates that GAPS is more likely to select the correct geometry elements when the enhancement is applied.

8.2.3 Improve the Quality of Fusing the Diagram and the Text Modality for Solving Geometry Maths Problems

In Chapter 6, we introduced the GOLD model, which aims to enhance the interpretation of geometric diagrams by representing them with natural language descriptions. The goal is to improve interpretability and establish connections between different modalities: geometric diagrams and problem texts. However, Chapter 5 highlighted the challenges in accurately encoding and interpreting geometric diagrams when jointly processing them with text. To address this issue, we state that using natural language descriptions for geometric diagrams can provide accurate and understandable representations while bridging the gap between textual and visual information (see **Statement (3)** in Section 1.3). To investigate this hypothesis, we addressed the research questions proposed in Section 6.1.

In Section 6.4.1, we assessed the effectiveness of our GOLD relative to current state-of-the-art methods. This comparison, as presented in the datasets UniGeo CAL, PRV, PGPS9K, and Geometry3K, demonstrated GOLD's superior performance, addressing Research Question **RQ-6.1**.

In Section 6.4.2, we delved into research question **RQ-6.2** by comparing the performance of our GOLD model using three distinct diagram description formats: absence of diagram description, logic forms, and natural language descriptions. Results displayed in Table 6.2 indicates that descriptions in natural language outperform those in logic-form on three LLMs: T5, Llama2, and CodeLlama.

To address research questions **RQ-6.3** and **RQ-6.4**, we conducted ablation studies. In the ablation study, as illustrated in Table 6.4 in Section 6.4.4, the extraction of both relation types reaches its highest F1-score when both embeddings (emb_{feat} and emb_{spat}) are utilised. Furthermore, in Section 6.4.5, we show that GOLD model using both embeddings performs the best on all test subsets.

We further address the research question **RQ-6.5** in Section 6.4.6, as illustrated in Figure 6.2, we noted a impact on GOLD's performance in the absence of the *geo2geo* relation. This finding confirms the critical role of the *geo2geo* relation in conveying essential details of geometric diagrams, answering **RQ-6.5**.

8.2.4 Evaluate Latest LLMs and MMs Proficiency in Geometry Maths Problem-Solving

From Chapter 6, we observe the advantage of using LLMs as problem solver, which generates the reasoning process. Additionally, we find that describing the geometric diagrams into natural language can improve models' performances. This raise our interest to investigate the latest LLMs and MMs on solving the geometry maths problems, where specific research on evaluating these models' effectiveness in geometry problem-solving remains scarce. We state that LLMs and MMs underperform on this task (see **Statment (4) in Section 1.3**). To support this statement, we introduce the GeoEval benchmark, a comprehensive collection specifically designed for this task. We aimed to validate this hypothesis by addressing specific research questions proposed in

Section 7.1.

In Chapter 4, we introduce the GeoEval benchmark, crafted specifically to evaluate the performance of LLMs and MMs in resolving geometry maths problems. This benchmark is meticulously created, featuring five key advantages: *Comprehensive Variety*, *Varied Problems*, *Dual Inputs*, *Diverse Challenges*, and *Complexity Ratings*. Utilising the proposed GeoEval benchmark, we assess the proficiency of ten state-of-the-art LLMs and MMs in solving geometry maths problems. Our analysis, as presented in Table 7.1, reveals that both LLMs and MMs continue to face difficulties in this domain. Notably, even the top-performing model, WizardMath-70B, achieves only 55.67% accuracy on the GeoEval-2000 subset, providing a response to Research Question **RQ-7.1**.

In Section 7.2.3.4, we address Research Question **RQ-7.2** by conducting an analysis of the highest-performing model on the GeoEval benchmark, WizardMath-70B. Our investigation leads us to the conclusion that models pre-trained on mathematical corpora exhibit superior performance in automated geometry maths problem-solving tasks. One notable advantage of these models is their implicit integration of the requisite mathematical knowledge necessary for solving geometry maths problems. However, despite the critical importance of pre-training on a mathematical corpus for tackling geometry maths problems, we also find that it may not suffice on its own.

In Section 7.2.4, we recognise the importance of integrating descriptions of geometric diagrams, which can notably enhance the ability of LLMs and MMs to comprehend and solve geometry problems. This significance is underscored by the results of the ablation study presented in Table 7.2, directly addressing Research Question **RQ-7.3**.

In addressing Research Question **RQ-7.4**, we conducted a comparative analysis of four representative models to assess their performances in solving geometry problems that either require external knowledge or do not. As depicted in Figure 7.1, we observed a decline in the performances of CodeGen2-16B, GPT-4, and GPT-4V on problems requiring external knowledge, with the exception of the WizardMath-7B-V1.1 model. Notably, WizardMath models [232] exhibit superior performance in automated geometry maths problem-solving tasks due to their inherent integration of the necessary mathematical knowledge essential for tackling geometry problems. This comprehensive evaluation directly addresses Research Question **RQ-7.4**.

In Section 7.2.5, we tackled both Research Question **RQ-7.5** and Research Question **RQ-7.6** by examining the performances of models concerning varying question lengths and complexity levels. As illustrated in Figure 7.2 and Figure 7.3, it is evident that the performance of models diminishes as both question lengths and complexity levels increase.

We further evaluated the proposed GAPS model discussed in Chapter 5 and the GOLD model presented in Chapter 6 on the GeoEval benchmark. The results shown in Table 5.5.6 and Table 6.4.3 demonstrate that GeoEval serves as a challenging benchmark for assessing the ability to solve geometry-based mathematical problems. The performance of the models on this benchmark highlights the complexity of the task and the need for robust approaches to tackle geometry problem-solving.

8.3 Future Work

In this thesis, we have explored how to enable AI agents to perform mathematical reasoning. While significant progress have been made, several directions remain open for further exploration and development. Future research could focus on the following areas:

8.3.1 Other Multi-Modal Mathematical Reasoning

In this thesis, we have delved into the design of methodologies for automated geometry problem solving, as elaborated in Chapters 5 and 6. Moving forward, it's crucial to consider other modalities such as tabular data and videos [19, 238] The method we proposed in Chapter 6, which involves converting geometric diagrams into natural language, shows promise for extending to multi-modal mathematical reasoning.

The future challenge lies in fully leveraging the rich information inherent in these diverse modalities and effectively utilising them for mathematical reasoning tasks. Potential areas for future research include:

- Exploring how to incorporate data from various sources such as videos and tables, and understanding how they can contribute to the problem-solving process in mathematical reasoning.

- Modifying the natural language translation approach of geometric diagrams, as proposed in Chapter 6, to suit other modalities. This involves developing new methods to interpret and translate complex information from different data formats into a comprehensible form for AI processing.

By addressing these areas, we might create AI systems capable of tackling complex mathematical reasoning tasks across a wider range of data formats.

8.3.2 Different Formats to Represent the Reasoning Process

In this thesis, we adopted solution programs as the representation medium for the reasoning process in mathematical reasoning. Despite the effectiveness of this approach, as shown in all chapters, a notable limitation emerges in Chapter 5: while GAPS demonstrates impressive performance in geometry proving problems, it does not show comparable results on geometry calculation problems. Our in-depth analysis revealed that GAPS excels when solution programs adhere to a specific pattern, a format where each operator is followed by the same number of operands. This finding opens up a promising direction for future research, which involves exploring more effective representation methods for the reasoning process.

Current work, such as using the natural language [18,36] and the programming language [100], each have their limitations. Natural language processing can inadvertently introduce noise, while programming language requires consuming conversion process. Therefore, we list potential future research directions as following:

- Developing a new representation method that combines the strengths of solution program natural language and programming language. This hybrid model could offer the clarity and specificity of solution program and programming language while retaining the flexibility and expressiveness of natural language.

- Enhancing the reasoning process representation by incorporating advanced contextual and semantic analysis techniques, which could help in understanding and solving more complex mathematical problems.

8.4 Chapter Summary

This chapter highlights key contributions and summarises findings of this thesis, which is identifying limitations in current methodologies and exploring automated mathematical reasoning using AI. It also proposes directions for future research, building on the groundwork from this thesis.

Bibliography

[1] J. Chen, T. Li, J. Qin, P. Lu, L. Lin, C. Chen, and X. Liang, "Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Association for Computational Linguistics, 2022, pp. 3313–3323. [Online]. Available: https://doi.org/10.18653/v1/2022.emnlp-main.218

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[3] Y. Shen and C. Jin, "Solving math word problems with multi-encoders and multi-decoders," in *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, D. Scott, N. Bel, and C. Zong, Eds. International Committee on Computational Linguistics, 2020, pp. 2924–2934. [Online]. Available: https://doi.org/10.18653/v1/2020.coling-main.262

[4] J. Cho, J. Lei, H. Tan, and M. Bansal, "Unifying vision-and-language tasks via text generation," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 1931–1942. [Online]. Available: http://proceedings.mlr.press/v139/cho21a.html

[5] M. Zhang, F. Yin, and C. Liu, "A multi-modal neural geometric solver with textual clauses parsed from diagram," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China.* ijcai.org, 2023, pp. 3374–3382. [Online]. Available: https://doi.org/10.24963/ijcai.2023/376

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[8] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: fully convolutional one-stage object detection," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019.* IEEE, 2019, pp. 9626–9635. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00972

[9] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. 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: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 2357–2367. [Online]. Available: https://doi.org/10.18653/v1/n19-1245

[10] Z. Chen, W. Chen, C. Smiley, S. Shah, I. Borova, D. Langdon, R. Moussa, M. Beane, T.-H. Huang, B. Routledge, and W. Y. Wang, "FinQA: A dataset of numerical reasoning over financial data," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3697–3711. [Online]. Available: https://aclanthology.org/2021.emnlp-main.300

[11] J. Chen, J. Tang, J. Qin, X. Liang, L. Liu, E. P. Xing, and L. Lin, "Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning," in *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, ser. Findings of ACL, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., vol. ACL/IJCNLP 2021. Association for Computational Linguistics, 2021, pp. 513–523. [Online]. Available: https://doi.org/10.18653/v1/2021.findings-acl.46

[12] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K. Chang, M. Galley, and J. Gao, "Mathvista: Evaluating math reasoning in visual contexts with gpt-4v, bard, and other large multimodal models," *CoRR*, vol. abs/2310.02255, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.02255

[13] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, 1950. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433

[14] P. J. Hayes and K. M. Ford, "Turing test considered harmful," in *Proceedings of the Fourteenth International Joint Conference on Artificial*

Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes. Morgan Kaufmann, 1995, pp. 972–977. [Online]. Available: http://ijcai.org/Proceedings/95-1/Papers/125.pdf

[15] P. Clark and O. Etzioni, "My computer is an honor student - but how intelligent is it? standardized tests as a measure of AI," *AI Mag.*, vol. 37, no. 1, pp. 5–12, 2016. [Online]. Available: https://doi.org/10.1609/aimag.v37i1.2636

[16] A. Bhattacharya, "A survey of question answering for math and science problem," *CoRR*, vol. abs/1705.04530, 2017. [Online]. Available: http://arxiv.org/abs/1705.04530

[17] S. Mishra, A. Mitra, N. Varshney, B. S. Sachdeva, P. Clark, C. Baral, and A. Kalyan, "Numglue: A suite of fundamental yet challenging mathematical reasoning tasks," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 3505–3523. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-long.246

[18] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the MATH dataset," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, J. Vanschoren and S. Yeung, Eds., 2021. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html

[19] P. Lu, L. Qiu, K. Chang, Y. N. Wu, S. Zhu, T. Rajpurohit, P. Clark, and A. Kalyan, "Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=DHyHRBwJUTN

[20] P. Lu, L. Qiu, W. Yu, S. Welleck, and K. Chang, "A survey of deep learning for mathematical reasoning," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, A. Rogers, J. L. Boyd-Graber, and N. Okazaki, Eds. Association for Computational Linguistics, 2023, pp. 14 605–14 631. [Online]. Available: https://doi.org/10.18653/v1/2023.acl-long.817

[21] D. G. Bobrow, "Natural language input for a computer problem solving system," 1964. [Online]. Available: https://api.semanticscholar.org/CorpusID:56584838

[22] A. V. Napalkov and Y. V. Orfeev, "Computers and thought, edited by e. a. feigenbaum and j. feldman, new york, mcgraw-hill, 1963: Book review,," 1967. [Online]. Available: https://api.semanticscholar.org/CorpusID:60228342

[23] E. Charniak, "Computer solution of calculus word problems," in *Proceedings of the 1st International Joint Conference on Artificial Intelligence, Washington, DC, USA, May 7-9, 1969*, D. E. Walker and L. M. Norton, Eds. William Kaufmann, 1969, pp. 303–316. [Online]. Available: http://ijcai.org/Proceedings/69/Papers/031.pdf

[24] D. J. Briars and J. H. Larkin, "An integrated model of skill in solving elementary word problems," *Cognition and instruction*, vol. 1, no. 3, pp. 245–296, 1984.

[25] C. R. Fletcher, "Understanding and solving arithmetic word problems: A computer simulation," *Behavior Research Methods, Instruments, & Computers*, vol. 17, no. 5, pp. 565–571, 1985.

[26] O. Tafjord, P. Clark, M. Gardner, W. Yih, and A. Sabharwal, "QUAREL: A dataset and models for answering questions about qualitative relationships," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 -*

February 1, 2019. AAAI Press, 2019, pp. 7063–7071. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33017063

[27] B. Zhou, D. Khashabi, Q. Ning, and D. Roth, "”going on a vacation” takes longer than ”going for a walk”: A study of temporal commonsense understanding," 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 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 3361–3367. [Online]. Available: https://doi.org/10.18653/v1/D19-1332

[28] A. Kalyan, A. Kumar, A. Chandrasekaran, A. Sabharwal, and P. Clark, "How much coffee was consumed during EMNLP 2019? fermi problems: A new reasoning challenge for AI," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 7318–7328. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-main.582

[29] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=H1gR5iR5FX

[30] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang, "Parsing algebraic word problems into equations," *Trans. Assoc. Comput. Linguistics*, vol. 3, pp. 585–597, 2015. [Online]. Available: https://doi.org/10.1162/tacl_a_00160

[31] A. Patel, S. Bhattamishra, and N. 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 2021, Online, June 6-11, 2021*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Association for Computational Linguistics, 2021, pp. 2080–2094. [Online]. Available: https://doi.org/10.18653/v1/2021.naacl-main.168

[32] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, "Learning to solve arithmetic word problems with verb categorization," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 523–533. [Online]. Available: https://doi.org/10.3115/v1/d14-1058

[33] N. Kushman, L. Zettlemoyer, R. Barzilay, and Y. Artzi, "Learning to automatically solve algebra word problems," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. The Association for Computer Linguistics, 2014, pp. 271–281. [Online]. Available: https://doi.org/10.3115/v1/p14-1026

[34] S. Roy, "Reasoning about quantities in natural language," Ph.D. dissertation, University of Illinois Urbana-Champaign, USA, 2017. [Online]. Available: https://hdl.handle.net/2142/98354

[35] "Ape210k: A large-scale and template-rich dataset of math word problems," *CoRR*, vol. abs/2009.11506, 2020, withdrawn. [Online]. Available: https://arxiv.org/abs/2009.11506

[36] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word

problems," *arXiv preprint arXiv:2110.14168*, 2021.

[37] S. E. Kahou, V. Michalski, A. Atkinson, Á. Kádár, A. Trischler, and Y. Bengio, "Figureqa: An annotated figure dataset for visual reasoning," *arXiv preprint arXiv:1710.07300*, 2017.

[38] K. Kafle, B. L. Price, S. Cohen, and C. Kanan, "DVQA: understanding data visualizations via question answering," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 5648–5656. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Kafle_DVQA_Understanding_Data_CVPR_2018_paper.html

[39] W. Zhang, C. Zhang, Y. Zhu, and S. Zhu, "Machine number sense: A dataset of visual arithmetic problems for abstract and relational reasoning," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 1332–1340. [Online]. Available: https://doi.org/10.1609/aaai.v34i02.5489

[40] B. Y. Lin, S. Lee, R. Khanna, and X. Ren, "Birds have four legs?! numersense: Probing numerical commonsense knowledge of pre-trained language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 6862–6868. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.557

[41] A. Ravichander, A. Naik, C. P. Rosé, and E. H. Hovy, "EQUATE: A benchmark evaluation framework for quantitative reasoning in natural language inference," in *Proceedings of the 23rd Conference on Computational Natural Language*

Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019, M. Bansal and A. Villavicencio, Eds. Association for Computational Linguistics, 2019, pp. 349–361. [Online]. Available: https://doi.org/10.18653/v1/K19-1033

[42] T. Schuster, A. Kalyan, A. Polozov, and A. Kalai, "Programming puzzles," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, J. Vanschoren and S. Yeung, Eds., 2021. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/3988c7f88ebcb58c6ce932b957b6f332-Abstract-round1.html

[43] M. J. Seo, H. Hajishirzi, A. Farhadi, O. Etzioni, and C. Malcolm, "Solving geometry problems: Combining text and diagram interpretation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, Eds. The Association for Computational Linguistics, 2015, pp. 1466–1476. [Online]. Available: https://doi.org/10.18653/v1/d15-1171

[44] C. Alvin, S. Gulwani, R. Majumdar, and S. Mukhopadhyay, "Synthesis of solutions for shaded area geometry problems," in *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2017, Marco Island, Florida, USA, May 22-24, 2017*, V. Rus and Z. Markov, Eds. AAAI Press, 2017, pp. 14–19. [Online]. Available: https://aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15416

[45] M. Sachan, A. Dubey, and E. P. 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, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, M. Palmer, R. Hwa, and S. Riedel, Eds. Association for Computational Linguistics, 2017, pp. 773–784. [Online]. Available: https://doi.org/10.18653/v1/d17-1081

[46] M. Sachan and E. P. 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 @ACM 2017, Vancouver, Canada, August 3-4, 2017*, N. Ide, A. Herbelot, and L. Màrquez, Eds. Association for Computational Linguistics, 2017, pp. 251–261. [Online]. Available: https://doi.org/10.18653/v1/S17-1029

[47] W. Yu, M. Wang, X. Wang, X. Zhou, Y. Zha, Y. Zhang, S. Miao, and J. Liu, "Geore: A relation extraction dataset for chinese geometry problems," in *35th Conference on Neural Information Processing Systems (NeurIPS 2021) Workshop on Math AI for Education (MATHAI4ED)*, 2021.

[48] A. Newell, J. C. Shaw, and H. A. Simon, "Empirical explorations of the logic theory machine: a case study in heuristic," in *Papers presented at the 1957 western joint computer conference: Techniques for reliability, IRE-AIEE-ACM 1957 (Western), Los Angeles, California, USA, February 26-28, 1957*, M. M. Astrahan, Ed. ACM, 1957, pp. 218–230. [Online]. Available: https://doi.org/10.1145/1455567.1455605

[49] S. Polu and I. Sutskever, "Generative language modeling for automated theorem proving," *CoRR*, vol. abs/2009.03393, 2020. [Online]. Available: https://arxiv.org/abs/2009.03393

[50] J. M. Han, J. Rute, Y. Wu, E. W. Ayers, and S. Polu, "Proof artifact co-training for theorem proving with language models," *arXiv preprint arXiv:2102.06203*, 2021.

[51] S. Polu, J. M. Han, K. Zheng, M. Baksys, I. Babuschkin, and I. Sutskever, "Formal mathematics statement curriculum learning," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=-P7G-8dmSh4

[52] A. Q. Jiang, W. Li, S. Tworkowski, K. Czechowski, T. Odrzygózdz, P. Milos, Y. Wu, and M. Jamnik, "Thor: Wielding hammers to integrate language models and automated theorem provers," in *NeurIPS*, 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/377c25312668e48f2e531e2f2c422483-Abstract-Conference.html

[53] G. Lample, T. Lacroix, M. Lachaux, A. Rodriguez, A. Hayat, T. Lavril, G. Ebner, and X. Martinet, "Hypertree proof search for neural theorem proving," in *NeurIPS*, 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/a8901c5e85fb8e1823bbf0f755053672-Abstract-Conference.html

[54] K. Yang and J. Deng, "Learning to prove theorems via interacting with proof assistants," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 6984–6994. [Online]. Available: http://proceedings.mlr.press/v97/yang19a.html

[55] A. Q. Jiang, W. Li, J. M. Han, and Y. Wu, "Lisa: Language models of isabelle proofs," in *6th Conference on Artificial Intelligence and Theorem Proving*, 2021, pp. 378–392.

[56] K. Zheng, J. M. Han, and S. Polu, "minif2f: a cross-system benchmark for formal olympiad-level mathematics," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.* OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=9ZPegFuFTFv

[57] Q. Wu, Q. Zhang, Z. Wei, and X. Huang, "Math word problem solving with explicit numerical values," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li,

and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 5859–5869. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.455

[58] X. Chen, C. Liang, A. W. Yu, D. Zhou, D. Song, and Q. V. Le, "Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=ryxjnREFwH

[59] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, "DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 2368–2378. [Online]. Available: https://doi.org/10.18653/v1/n19-1246

[60] P. Lu, R. Gong, S. Jiang, L. Qiu, S. Huang, X. Liang, and S. Zhu, "Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 6774–6786. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.528

[61] J. Cao and J. Xiao, "An augmented benchmark dataset for geometric question answering through dual parallel text encoding," in *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, N. Calzolari, C. Huang, H. Kim, J. Pustejovsky, L. Wanner, K. Choi, P. Ryu, H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm,

Z. He, T. K. Lee, E. Santus, F. Bond, and S. Na, Eds. International Committee on Computational Linguistics, 2022, pp. 1511–1520. [Online]. Available: https://aclanthology.org/2022.coling-1.130

[62] M. Zhang, F. Yin, Y. Hao, and C. Liu, "Plane geometry diagram parsing," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. D. Raedt, Ed. ijcai.org, 2022, pp. 1636–1643. [Online]. Available: https://doi.org/10.24963/ijcai.2022/228

[63] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12346. Springer, 2020, pp. 213–229. [Online]. Available: https://doi.org/10.1007/978-3-030-58452-8_13

[64] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[65] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International conference on machine learning*. PMLR, 2018, pp. 4055–4064.

[66] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 2021, pp. 5904–5908. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9413535

[67] L. Dong, S. Xu, and B. Xu, “Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018.* IEEE, 2018, pp. 5884–5888. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8462506

[68] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, “Conformer: Convolution-augmented transformer for speech recognition,” in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 5036–5040. [Online]. Available: https://doi.org/10.21437/Interspeech.2020-3015

[69] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 8748–8763. [Online]. Available: http://proceedings.mlr.press/v139/radford21a.html

[70] J. Li, D. Li, C. Xiong, and S. C. H. Hoi, “BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation,” in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 12 888–12 900. [Online]. Available: https://proceedings.mlr.press/v162/li22n.html

[71] H. Bao, L. Dong, S. Piao, and F. Wei, “Beit: BERT pre-training of image transformers,” in *The Tenth International Conference on Learning*

Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=p-BhZSz59o4

[72] J. Li, R. R. Selvaraju, A. Gotmare, S. R. Joty, C. Xiong, and S. C. Hoi, “Align before fuse: Vision and language representation learning with momentum distillation,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 9694–9705. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/505259756244493872b7709a8a01b536-Abstract.html

[73] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *CoRR*, vol. abs/2001.08361, 2020. [Online]. Available: https://arxiv.org/abs/2001.08361

[74] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[75] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “Llama: Open and efficient foundation language models,” *CoRR*, vol. abs/2302.13971, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.13971

[76] OpenAI, "GPT-4 technical report," *CoRR*, vol. abs/2303.08774, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.08774

[77] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016. [Online]. Available: http://arxiv.org/abs/1607.06450

[78] J. Devlin, M. Chang, K. Lee, and K. 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 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423

[79] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[80] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtvS

[81] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[82] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A conditional transformer language model for controllable generation," *CoRR*, vol. abs/1909.05858, 2019. [Online]. Available: http://arxiv.org/abs/1909.05858

[83] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[84] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. [Online]. Available: https://aclanthology.org/P19-1285

[85] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. 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*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[86] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, "Multilingual denoising pre-training for neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 726–742, 2020. [Online]. Available: https://aclanthology.org/2020.tacl-1.47

[87] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[88] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy, "The pile: An 800gb

dataset of diverse text for language modeling," *CoRR*, vol. abs/2101.00027, 2021. [Online]. Available: https://arxiv.org/abs/2101.00027

[89] Z. Li, X. Ding, and T. Liu, "Story ending prediction by transferable BERT," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 1800–1806. [Online]. Available: https://doi.org/10.24963/ijcai.2019/249

[90] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune BERT for text classification?" in *Chinese Computational Linguistics - 18th China National Conference, CCL 2019, Kunming, China, October 18-20, 2019, Proceedings*, ser. Lecture Notes in Computer Science, M. Sun, X. Huang, H. Ji, Z. Liu, and Y. Liu, Eds., vol. 11856. Springer, 2019, pp. 194–206. [Online]. Available: https://doi.org/10.1007/978-3-030-32381-3_16

[91] S. Gururangan, A. Marasovic, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 8342–8360. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.740

[92] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *CoRR*, vol. abs/2001.08361, 2020. [Online]. Available: https://arxiv.org/abs/2001.08361

[93] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, "Training compute-optimal

large language models," *CoRR*, vol. abs/2203.15556, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2203.15556

[94] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," *J. Mach. Learn. Res.*, vol. 24, pp. 240:1–240:113, 2023. [Online]. Available: http://jmlr.org/papers/v24/22-1144.html

[95] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilic, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, J. Tow, A. M. Rush, S. Biderman, A. Webson, P. S. Ammanamanchi, T. Wang, B. Sagot, N. Muennighoff, A. V. del Moral, O. Ruwase, R. Bawden, S. Bekman, A. McMillan-Major, I. Beltagy, H. Nguyen, L. Saulnier, S. Tan, P. O. Suarez, V. Sanh, H. Laurençon, Y. Jernite, J. Launay, M. Mitchell, C. Raffel, A. Gokaslan, A. Simhi, A. Soroa, A. F. Aji, A. Alfassy, A. Rogers, A. K. Nitzav, C. Xu, C. Mou, C. Emezue, C. Klamm, C. Leong, D. van Strien, D. I. Adelani, and et al., "BLOOM: A 176b-parameter open-access multilingual language model," *CoRR*, vol. abs/2211.05100, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.05100

[96] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," *CoRR*, vol. abs/2211.09085, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.09085

[97] J. Manyika, "An overview of bard: an early experiment with generative ai," *AI. Google Static Documents*, 2023.

[98] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *NeurIPS*, 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[99] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=1PL1NIMMrw

[100] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "PAL: program-aided language models," in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 2023, pp. 10 764–10 799. [Online]. Available: https://proceedings.mlr.press/v202/gao23f.html

[101] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, and E. H. Chi, "Least-to-most prompting enables complex reasoning in large language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=WZH7099tgfM

[102] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J. Lou, and W. Chen, "On the advance of making language models better reasoners," *CoRR*, vol. abs/2206.02336, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2206.02336

[103] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal, "Decomposed prompting: A modular approach for solving complex tasks," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=_nGgzQjzaRy

[104] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *CoRR*, vol. abs/2211.12588, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.12588

[105] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, M. Palmer, R. Hwa, and S. Riedel, Eds. Association for Computational Linguistics, 2017, pp. 845–854. [Online]. Available: https://doi.org/10.18653/v1/d17-1088

[106] J. Qin, L. Lin, X. Liang, R. Zhang, and L. 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 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 3780–3789. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.309

[107] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom, "Program induction by rationale generation: Learning to solve and explain algebraic word problems," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 158–167. [Online]. Available: https://doi.org/10.18653/v1/P17-1015

[108] S. Mishra, M. Finlayson, P. Lu, L. Tang, S. Welleck, C. Baral, T. Rajpurohit, O. Tafjord, A. Sabharwal, P. Clark *et al.*, "Lila: A unified benchmark for mathematical reasoning," *arXiv preprint arXiv:2210.17517*, 2022.

[109] F. Zhu, W. Lei, Y. Huang, C. Wang, S. Zhang, J. Lv, F. Feng, and T. 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/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 3277–3287. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.254

[110] Y. Zhao, Y. Li, C. Li, and R. Zhang, "Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 6588–6600. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-long.454

[111] S. Upadhyay and M. 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, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, M. Lapata, P. Blunsom, and A. Koller, Eds. Association for Computational Linguistics, 2017, pp. 494–504. [Online]. Available: https://doi.org/10.18653/v1/e17-1047

[112] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, "MAWPS: A math word problem repository," in *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, K. Knight, A. Nenkova, and O. Rambow, Eds. The Association

for Computational Linguistics, 2016, pp. 1152–1157. [Online]. Available: https://doi.org/10.18653/v1/n16-1136

[113] D. Huang, S. Shi, C. Lin, J. Yin, and W. 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 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers.* The Association for Computer Linguistics, 2016. [Online]. Available: https://doi.org/10.18653/v1/p16-1084

[114] S. Roy and D. Roth, "Unit dependency graph and its application to arithmetic word problem solving," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, S. Singh and S. Markovitch, Eds. AAAI Press, 2017, pp. 3082–3088. [Online]. Available: https://doi.org/10.1609/aaai.v31i1.10959

[115] ——, "Mapping to declarative knowledge for word problem solving," *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 159–172, 2018. [Online]. Available: https://doi.org/10.1162/tacl_a_00012

[116] S. Miao, C. Liang, and K. 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 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 975–984. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.92

[117] J. Austin, A. Odena, M. I. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. J. Cai, M. Terry, Q. V. Le, and C. Sutton, "Program synthesis with large language models," *CoRR*, vol. abs/2108.07732, 2021. [Online]. Available: https://arxiv.org/abs/2108.07732

[118] E. Charniak, "Calculus word problems," Ph.D. dissertation, Ph. D. thesis, Massachusetts Institute of Technology, 1968.

[119] D. Dellarosa, "A computer simulation of children's arithmetic word-problem solving," *Behavior Research Methods, Instruments, & Computers*, vol. 18, no. 2, pp. 147–154, 1986.

[120] Y. Bakman, "Robust understanding of word problems with extraneous information," *arXiv preprint math/0701393*, 2007.

[121] M. Yuhui, Z. Ying, C. Guangzuo, R. Yun, and H. Ronghuai, "Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems," in *2010 Second International Workshop on Education Technology and Computer Science*, vol. 2. IEEE, 2010, pp. 476–479.

[122] L. Zhou, S. Dai, and L. Chen, "Learn to solve algebra word problems using quadratic programming," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, Eds. The Association for Computational Linguistics, 2015, pp. 817–822. [Online]. Available: https://doi.org/10.18653/v1/d15-1096

[123] A. Mitra and C. Baral, "Learning to use formulas to solve simple arithmetic problems," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. [Online]. Available: https://doi.org/10.18653/v1/p16-1202

[124] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 3104–3112. [Online]. Available: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

[125] G. Tiwari, A. Sharma, A. Sahotra, and R. Kapoor, "English-hindi neural machine translation-lstm seq2seq and convs2s," in *2020 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2020, pp. 871–875.

[126] X. He, G. Haffari, and M. Norouzi, "Sequence to sequence mixture model for diverse machine translation," in *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, A. Korhonen and I. Titov, Eds. Association for Computational Linguistics, 2018, pp. 583–592. [Online]. Available: https://doi.org/10.18653/v1/k18-1056

[127] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *Trans. Data Sci.*, vol. 2, no. 1, pp. 1:1–1:37, 2021. [Online]. Available: https://doi.org/10.1145/3419106

[128] S. Xu, X. Zhang, Y. Wu, and F. Wei, "Sequence level contrastive learning for text summarization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 10, 2022, pp. 11 556–11 565.

[129] C. Liu, F. Sun, C. Wang, F. Wang, and A. L. Yuille, "MAT: A multimodal attentive translator for image captioning," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, C. Sierra, Ed. ijcai.org, 2017, pp. 4033–4039. [Online]. Available: https://doi.org/10.24963/ijcai.2017/563

[130] M. Yang, J. Liu, Y. Shen, Z. Zhao, X. Chen, Q. Wu, and C. Li, "An ensemble of generation-and retrieval-based image captioning with dual generator generative adversarial network," *IEEE Transactions on Image Processing*, vol. 29, pp. 9627–9640, 2020.

[131] A. Fan, M. Lewis, and Y. N. Dauphin, "Hierarchical neural story generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, Eds. Association

for Computational Linguistics, 2018, pp. 889–898. [Online]. Available: https://aclanthology.org/P18-1082/

[132] L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao, and R. Yan, "Plan-and-write: Towards better automatic storytelling," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7378–7385.

[133] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[134] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1724–1734. [Online]. Available: https://doi.org/10.3115/v1/d14-1179

[135] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[136] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a math word problem to an expression tree," *CoRR*, vol. abs/1811.05632, 2018. [Online]. Available: http://arxiv.org/abs/1811.05632

[137] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. T. Dai, and H. T. Shen, "Template-based math word problem solvers with recursive neural networks," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 7144–7151. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33017144

[138] T. Chiang and Y. Chen, “Semantically-aligned equation generation for solving and reasoning math word problems,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 2656–2668. [Online]. Available: https://doi.org/10.18653/v1/n19-1272

[139] J. Li, L. Wang, J. Zhang, Y. Wang, B. T. Dai, and D. Zhang, “Modeling intra-relation in math word problems with different functional multi-head attentions,” in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 6162–6167. [Online]. Available: https://doi.org/10.18653/v1/p19-1619

[140] Y. Hong, Q. Li, R. Gong, D. Ciao, S. Huang, and S. Zhu, “SMART: A situation model for algebra story problems via attributed grammar,” in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* AAAI Press, 2021, pp. 13 009–13 017. [Online]. Available: https://doi.org/10.1609/aaai.v35i14.17538

[141] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom, “Program induction by rationale generation: Learning to solve and explain algebraic word problems,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 158–167. [Online]. Available: https://doi.org/10.18653/v1/P17-1015

[142] B. Robaidek, R. Koncel-Kedziorski, and H. Hajishirzi, “Data-driven methods for solving algebra word problems,” *CoRR*, vol. abs/1804.10718, 2018. [Online].

Available: http://arxiv.org/abs/1804.10718

[143] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[144] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 2692–2700. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html

[145] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 1073–1083. [Online]. Available: https://doi.org/10.18653/v1/P17-1099

[146] Z. Xie and S. 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 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 5299–5305. [Online]. Available: https://doi.org/10.24963/ijcai.2019/736

[147] Q. Liu, W. Guan, S. Li, and D. Kawahara, "Tree-structured decoding for solving math word problems," 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 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan,

Eds. Association for Computational Linguistics, 2019, pp. 2370–2379. [Online]. Available: https://doi.org/10.18653/v1/D19-1241

[148] K. Yang and J. Deng, "Learning to prove theorems via interacting with proof assistants," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 6984–6994. [Online]. Available: http://proceedings.mlr.press/v97/yang19a.html

[149] K. Zaporojets, G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, "Solving arithmetic word problems by scoring equations with recursive neural networks," *Expert Syst. Appl.*, vol. 174, p. 114704, 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.114704

[150] D. Huang, J. Liu, C. Lin, and J. Yin, "Neural math word problem solver with reinforcement learning," in *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, E. M. Bender, L. Derczynski, and P. Isabelle, Eds. Association for Computational Linguistics, 2018, pp. 213–223. [Online]. Available: https://aclanthology.org/C18-1018/

[151] Q. Wu, Q. Zhang, J. Fu, and X. Huang, "A knowledge-aware sequence-to-tree network for math word problem solving," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 7137–7146. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.579

[152] Q. Wu, Q. Zhang, Z. Wei, and X. Huang, "Math word problem solving with explicit numerical values," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1:*

Long Papers), Virtual Event, August 1-6, 2021, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 5859–5869. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.455

[153] Y. Hong, Q. Li, D. Ciao, S. Huang, and S. Zhu, "Learning by fixing: Solving math word problems with weak supervision," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* AAAI Press, 2021, pp. 4959–4967. [Online]. Available: https://doi.org/10.1609/aaai.v35i6.16629

[154] J. Qin, X. Liang, Y. Hong, J. Tang, and L. Lin, "Neural-symbolic solver for math word problems with auxiliary tasks," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 5870–5881. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.456

[155] Y. Cao, F. Hong, H. Li, and P. Luo, "A bottom-up DAG structure extraction model for math word problems," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* AAAI Press, 2021, pp. 39–46. [Online]. Available: https://doi.org/10.1609/aaai.v35i1.16075

[156] X. Lin, Z. Huang, H. Zhao, E. Chen, Q. Liu, H. Wang, and S. Wang, "HMS: A hierarchical solver with dependency-enhanced understanding for math word problem," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial*

Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. AAAI Press, 2021, pp. 4232–4240. [Online]. Available: https://doi.org/10.1609/aaai.v35i5.16547

[157] Q. Wu, Q. Zhang, and Z. Wei, "An edge-enhanced hierarchical graph-to-tree network for math word problem solving," in *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 1473–1482. [Online]. Available: https://doi.org/10.18653/v1/2021.findings-emnlp.127

[158] J. Zhang, L. Wang, R. K. Lee, Y. Bin, Y. Wang, J. Shao, and E. Lim, "Graph-to-tree learning for solving math word problems," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 3928–3937. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.362

[159] B. Kim, K. S. Ki, D. Lee, and G. Gweon, "Point to the expression: Solving algebraic word problems using the expression-pointer transformer model," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 3768–3779. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.308

[160] W. Yu, Y. Wen, F. Zheng, and N. Xiao, "Improving math word problems with pre-trained knowledge and hierarchical reasoning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 3384–3394. [Online]. Available: https://doi.org/10.18653/v1/2021.emnlp-main.272

[161] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.* OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[162] S. Welleck, J. Liu, X. Lu, H. Hajishirzi, and Y. Choi, “Naturalprover: Grounded mathematical proof generation with language models,” in *NeurIPS*, 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/1fc548a8243ad06616eee731e0572927-Abstract-Conference.html

[163] J. Shen, Y. Yin, L. Li, L. Shang, X. Jiang, M. Zhang, and Q. Liu, “Generate & rank: A multi-task framework for math word problems,” in *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 2269–2279. [Online]. Available: https://doi.org/10.18653/v1/2021.findings-emnlp.195

[164] A. Ni, J. P. Inala, C. Wang, O. Polozov, C. Meek, D. R. Radev, and J. Gao, “Learning from self-sampled correct and partially-correct programs,” *CoRR*, vol. abs/2205.14318, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.14318

[165] Z. Li, W. Zhang, C. Yan, Q. Zhou, C. Li, H. Liu, and Y. Cao, “Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems,” in *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 2486–2496. [Online]. Available: https://doi.org/10.18653/v1/2022.findings-acl.195

[166] M. I. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, C. Sutton, and A. Odena, "Show your work: Scratchpads for intermediate computation with language models," *CoRR*, vol. abs/2112.00114, 2021. [Online]. Available: https://arxiv.org/abs/2112.00114

[167] Z. Jie, J. Li, and W. Lu, "Learning to reason deductively: Math word problem solving as complex relation extraction," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 5944–5955. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-long.410

[168] M. Geva, A. Gupta, and J. Berant, "Injecting numerical reasoning skills into language models," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 946–958. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.89

[169] C. S. Peirce, *Reasoning and the logic of things: The Cambridge conferences lectures of 1898*. Harvard University Press, 1992.

[170] Y. Wu, M. N. Rabe, W. Li, J. Ba, R. B. Grosse, and C. Szegedy, "LIME: learning inductive bias for primitives of mathematical reasoning," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 11 251–11 262. [Online]. Available: http://proceedings.mlr.press/v139/wu21c.html

[171] Z. Liang, J. Zhang, L. Wang, W. Qin, Y. Lan, J. Shao, and X. Zhang, "MWP-BERT: numeracy-augmented pre-training for math word problem solving," in *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle,*

WA, United States, July 10-15, 2022, M. Carpuat, M. de Marneffe, and I. V. M. Ruíz, Eds. Association for Computational Linguistics, 2022, pp. 997–1009. [Online]. Available: https://doi.org/10.18653/v1/2022.findings-naacl.74

[172] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J. Lou, "TAPEX: table pre-training via learning a neural SQL executor," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.* OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=O50443AsCP

[173] A. Q. Jiang, W. Li, S. Tworkowski, K. Czechowski, T. Odrzygózdz, P. Milos, Y. Wu, and M. Jamnik, "Thor: Wielding hammers to integrate language models and automated theorem provers," in *NeurIPS*, 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/377c25312668e48f2e531e2f2c422483-Abstract-Conference.html

[174] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra, "Solving quantitative reasoning problems with language models," in *NeurIPS*, 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html

[175] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. S. Rosenberg, and G. Mann, "Bloomberggpt: A large language model for finance," *CoRR*, vol. abs/2303.17564, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.17564

[176] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the role of demonstrations: What makes in-context learning work?" in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds.

Association for Computational Linguistics, 2022, pp. 11 048–11 064. [Online]. Available: https://doi.org/10.18653/v1/2022.emnlp-main.759

[177] J. Piaget, *Child's Conception of Number: Selected Works vol 2.* Routledge, 2013.

[178] O. Rubin, J. Herzig, and J. Berant, "Learning to retrieve prompts for in-context learning," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, M. Carpuat, M. de Marneffe, and I. V. M. Ruíz, Eds. Association for Computational Linguistics, 2022, pp. 2655–2671. [Online]. Available: https://doi.org/10.18653/v1/2022.naacl-main.191

[179] Z. Zhang, A. Zhang, M. Li, and A. Smola, "Automatic chain of thought prompting in large language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=5NTt8GFjUHkr

[180] W. Yu, D. Iter, S. Wang, Y. Xu, M. Ju, S. Sanyal, C. Zhu, M. Zeng, and M. Jiang, "Generate rather than retrieve: Large language models are strong context generators," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=fB0hRu9GZUS

[181] Y. Fu, H. Peng, A. Sabharwal, P. Clark, and T. Khot, "Complexity-based prompting for multi-step reasoning," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=yf1icZHC-l9

[182] Z. Liang, W. Yu, T. Rajpurohit, P. Clark, X. Zhang, and A. Kalyan, "Let GPT be a math tutor: Teaching math word problem solvers with customized exercise generation," *CoRR*, vol. abs/2305.14386, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.14386

[183] Y. Hao, M. Zhang, F. Yin, and L. Huang, “PGDP5K: A diagram parsing dataset for plane geometry problems,” in *26th International Conference on Pattern Recognition, ICPR 2022, Montreal, QC, Canada, August 21-25, 2022.* IEEE, 2022, pp. 1763–1769. [Online]. Available: https://doi.org/10.1109/ICPR56361.2022.9956397

[184] H. L. Gelernter, J. R. Hansen, and D. W. Loveland, “Empirical explorations of the geometry theorem machine,” in *Papers presented at the 1960 western joint IRE-AIEE-ACM computer conference, IRE-AIEE-ACM 1960 (Western), San Francisco, California, USA, May 3-5, 1960*, R. M. Bennett, Ed. ACM, 1960, pp. 143–149. [Online]. Available: https://doi.org/10.1145/1460361.1460381

[185] W. Wen-Tsün, “Basic principles of mechanical theorem proving in elementary geometries,” *J. Autom. Reason.*, vol. 2, no. 3, pp. 221–252, 1986. [Online]. Available: https://doi.org/10.1007/BF02328447

[186] S. Chou and X. Gao, “Automated generation of readable proofs with geometric invariants i. multiple and shortest proof generation,” *J. Autom. Reason.*, vol. 17, no. 3, pp. 325–347, 1996. [Online]. Available: https://doi.org/10.1007/BF00283133

[187] Z. Ye, S. Chou, and X. Gao, “An introduction to java geometry expert - (extended abstract),” in *Automated Deduction in Geometry - 7th International Workshop, ADG 2008, Shanghai, China, September 22-24, 2008. Revised Papers*, ser. Lecture Notes in Computer Science, T. Sturm and C. Zengler, Eds., vol. 6301. Springer, 2008, pp. 189–195. [Online]. Available: https://doi.org/10.1007/978-3-642-21046-4_10

[188] M. Chinnappan, “Schemas and mental models in geometry problem solving,” *Educational Studies in Mathematics*, vol. 36, pp. 201–217, 1998.

[189] A. S. Nur and E. Nurvitasari, “Geometry skill analysis in problem solving reviewed from the difference of cognitive style students junior high school,” *Journal of Educational Science and Technology*, vol. 3, no. 3, pp. 204–210, 2017.

[190] J. M. Zelle and R. J. Mooney, "Learning semantic grammars with constructive inductive logic programming," in *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993*, R. Fikes and W. G. Lehnert, Eds. AAAI Press / The MIT Press, 1993, pp. 817–822. [Online]. Available: http://www.aaai.org/Library/AAAI/1993/aaai93-122.php

[191] ——, "Learning to parse database queries using inductive logic programming," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, W. J. Clancey and D. S. Weld, Eds. AAAI Press / The MIT Press, 1996, pp. 1050–1055. [Online]. Available: http://www.aaai.org/Library/AAAI/1996/aaai96-156.php

[192] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017. [Online]. Available: https://doi.org/10.1109/TPAMI.2016.2577031

[193] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2980–2988. [Online]. Available: https://doi.org/10.1109/ICCV.2017.322

[194] J. Kim, J. Jun, and B. Zhang, "Bilinear attention networks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 1571–1581. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/96ea64f3a1aa2fd00c72faacf0cb8ac9-Abstract.html

[195] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the*

Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, S. A. McIlraith and K. Q. Weinberger, Eds. AAAI Press, 2018, pp. 3942–3951. [Online]. Available: https://doi.org/10.1609/aaai.v32i1.11671

[196] P. Gao, Z. Jiang, H. You, P. Lu, S. C. H. Hoi, X. Wang, and H. Li, "Dynamic fusion with intra- and inter-modality attention flow for visual question answering," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 6639–6648. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Gao_Dynamic_Fusion_With_Intra-_and_Inter-Modality_Attention_Flow_for_Visual_CVPR_2019_paper.html

[197] M. Ning, Q. Wang, K. Huang, and X. Huang, "A symbolic characters aware model for solving geometry problems," in *Proceedings of the 31st ACM International Conference on Multimedia, MM 2023, Ottawa, ON, Canada, 29 October 2023- 3 November 2023*, A. El-Saddik, T. Mei, R. Cucchiara, M. Bertini, D. P. T. Vallejo, P. K. Atrey, and M. S. Hossain, Eds. ACM, 2023, pp. 7767–7775. [Online]. Available: https://doi.org/10.1145/3581783.3612570

[198] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 5754–5764. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html

[199] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the*

2016 Conference on Empirical Methods in Natural Language Processing, J. Su, K. Duh, and X. Carreras, Eds. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. [Online]. Available: https://aclanthology.org/D16-1264

[200] J. Zhang, R. K. Lee, E. Lim, W. Qin, L. Wang, J. Shao, and Q. Sun, "Teacher-student networks with multiple decoders for solving math word problem," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed. ijcai.org, 2020, pp. 4011–4017. [Online]. Available: https://doi.org/10.24963/ijcai.2020/555

[201] Q. Ran, Y. Lin, P. Li, J. Zhou, and Z. Liu, "NumNet: Machine reading comprehension with numerical reasoning," 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)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2474–2484. [Online]. Available: https://aclanthology.org/D19-1251

[202] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1024–1034. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[203] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019,*

December 8-14, 2019, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[204] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *CoRR*, vol. abs/1910.03771, 2019. [Online]. Available: http://arxiv.org/abs/1910.03771

[205] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[206] B. M. Lake and M. Baroni, "Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 2879–2888. [Online]. Available: http://proceedings.mlr.press/v80/lake18a.html

[207] X. Zhang, C. Li, Y. Zong, Z. Ying, L. He, and X. Qiu, "Evaluating the performance of large language models on GAOKAO benchmark," *CoRR*, vol. abs/2305.12474, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.12474

[208] S. Chou and X. Gao, "Automated generation of readable proofs with geometric invariants i. multiple and shortest proof generation," *J. Autom. Reason.*, vol. 17, no. 3, pp. 325–347, 1996.

[209] Z. Ye, S. Chou, and X. Gao, "An introduction to java geometry expert - (extended abstract)," in *Automated Deduction in Geometry - 7th International Workshop,*

ADG 2008, Shanghai, China, September 22-24, 2008. Revised Papers, ser. Lecture Notes in Computer Science, T. Sturm and C. Zengler, Eds., vol. 6301. Springer, 2008, pp. 189–195.

[210] M. Zhang, Z. Li, F. Yin, and C. Liu, "LANS: A layout-aware neural solver for plane geometry problem," *CoRR*, vol. abs/2311.16476, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2311.16476

[211] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, "Solving olympiad geometry without human demonstrations," *Nature*, vol. 625, no. 7995, pp. 476–482, 2024.

[212] J. Zhang, Y. Jiang, and Y. Moshfeghi, "GAPS: geometry-aware problem solver," *CoRR*, vol. abs/2401.16287, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.16287

[213] P. Lu, R. Gong, S. Jiang, L. Qiu, S. Huang, X. Liang, and S. Zhu, "Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 6774–6786. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-long.528

[214] M. Kazemi, H. Alvari, A. Anand, J. Wu, X. Chen, and R. Soricut, "Geomverse: A systematic evaluation of large models for geometric reasoning," *CoRR*, vol. abs/2312.12241, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2312.12241

[215] S. Tsai, C. Liang, H. Wang, and K. Su, "Sequence to general tree: Knowledge-guided geometry word problem solving," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August*

1-6, 2021, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 964–972. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-short.121

[216] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the MATH dataset," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, J. Vanschoren and S. Yeung, Eds., 2021. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html

[217] W. Jiang, H. Shi, L. Yu, Z. Liu, Y. Zhang, Z. Li, and J. T. Kwok, "Forward-backward reasoning in large language models for mathematical verification," 2023.

[218] L. Yu, W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu, "Metamath: Bootstrap your own mathematical questions for large language models," *CoRR*, vol. abs/2309.12284, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2309.12284

[219] OpenAI, "gpt-3.5-turbo-0125," 2022. [Online]. Available: https://openai.com/blog/chatgpt

[220] J. Zhang and Y. Moshfeghi, "Elastic: Numerical reasoning with adaptive symbolic compiler," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 12 647–12 661. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/522ef98b1e52f5918e5abc868651175d-Paper-Conference.pdf

[221] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. W. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances in Neural Information Processing Systems 30: Annual*

Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4967–4976. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/e6acf4b0f69f6f6e60e9a815938aa1ff-Abstract.html

[222] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian, "Deep modular co-attention networks for visual question answering," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019.* Computer Vision Foundation / IEEE, 2019, pp. 6281–6290. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Yu_Deep_Modular_Co-Attention_Networks_for_Visual_Question_Answering_CVPR_2019_paper.html

[223] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017.* IEEE Computer Society, 2017, pp. 936–944. [Online]. Available: https://doi.org/10.1109/CVPR.2017.106

[224] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018.* Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4510–4520. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[225] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. Canton-Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve, "Code llama: Open foundation models for code," *CoRR*, vol. abs/2308.12950, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.12950

[226] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *CoRR*, vol. abs/2211.12588, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.12588

[227] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[228] Z. Zhang, A. Zhang, M. Li, H. Zhao, G. Karypis, and A. Smola, "Multimodal chain-of-thought reasoning in language models," *CoRR*, vol. abs/2302.00923, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.00923

[229] T. Sawada, D. Paleka, A. Havrilla, P. Tadepalli, P. Vidas, A. Kranias, J. J. Nay, K. Gupta, and A. Komatsuzaki, "ARB: advanced reasoning benchmark for large language models," *CoRR*, vol. abs/2307.13692, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.13692

[230] O. Sainz, J. A. Campos, I. García-Ferrero, J. Etxaniz, O. L. de Lacalle, and E. Agirre, "NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark," in *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, 2023, pp. 10 776–10 787. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.722

[231] E. Nijkamp, H. Hayashi, C. Xiong, S. Savarese, and Y. Zhou, "Codegen2: Lessons for training llms on programming and natural languages," *CoRR*, vol.

abs/2305.02309, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.02309

[232] H. Luo, Q. Sun, C. Xu, P. Zhao, J. Lou, C. Tao, X. Geng, Q. Lin, S. Chen, and D. Zhang, "Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct," *CoRR*, vol. abs/2308.09583, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.09583

[233] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," *CoRR*, vol. abs/2310.03744, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.03744

[234] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou, "Qwen-vl: A frontier large vision-language model with versatile abilities," *arXiv preprint arXiv:2308.12966*, 2023. [Online]. Available: https://arxiv.org/abs/2308.12966

[235] Q. Ye, H. Xu, J. Ye, M. Yan, A. Hu, H. Liu, Q. Qian, J. Zhang, F. Huang, and J. Zhou, "mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration," *CoRR*, vol. abs/2311.04257, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2311.04257

[236] B. Peng, C. Li, P. He, M. Galley, and J. Gao, "Instruction tuning with GPT-4," *CoRR*, vol. abs/2304.03277, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.03277

[237] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen technical report," *CoRR*, vol. abs/2309.16609, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2309.16609

[238] P. Lu, L. Qiu, J. Chen, T. Xia, Y. Zhao, W. Zhang, Z. Yu, X. Liang, and S. Zhu, "Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, J. Vanschoren and S. Yeung, Eds., 2021. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/d3d9446802a44259755d38e6d163e820-Abstract-round2.html

[239] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 779–788. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[240] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9905. Springer, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007/978-3-319-46448-0_2

[241] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation with a discriminative loss function," *CoRR*, vol. abs/1708.02551, 2017. [Online]. Available: http://arxiv.org/abs/1708.02551

Appendix A

ELASTIC: Numerical Reasoning with Adaptive Symbolic Compiler

A.1 Example for a Maths Word Problems

Figure A.1 shows an example of a maths problem and its reasoning programs. The problem text P and question text Q are combined, which contains three numbers: 5, 3, 2, denoted by NUM. The reasoning program R contains two sub-programs, "$5 + 3$" (denoted as r_1) and "#0 $-$ 2" (denoted as r_2). The first sub-program r_1 contains one operator "+" (denoted as op_1) and two operands "5" (denoted as oe_1^1) and "3" (denoted as oe_2^1). The second sub-program r_2 contains one operator "$-$" (denoted as op_2) and two operands "#0" (denoted as oe_1^2) and "2" (denoted as oe_2^2). "#0" is pre-defined in the constant vocabulary, which is denoted as $CONS$. The op_1 and op_2 are belong to all mathematical operators OP. The oe_1^1, oe_2^1, oe_1^2, and oe_2^2 belong to all operands OE. We regard OP, OE as symbols s.

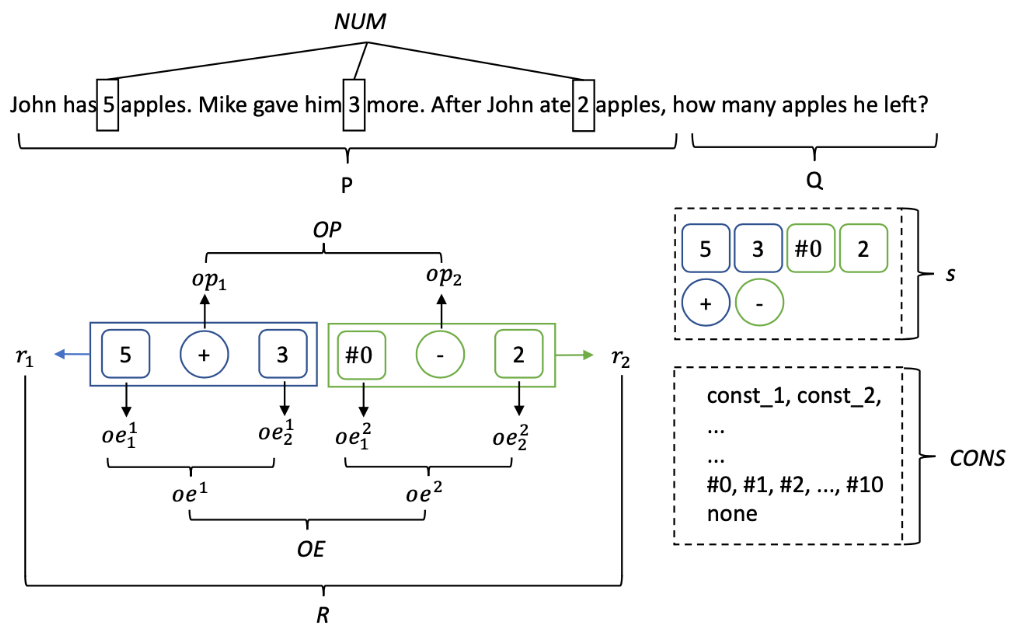


Figure A.1: An example of maths word problems, using nations defined in Table 3.1.

A.2 An Example to Show How Separated Modules Work

Figure A.2 shows the generation process of equations: “$5+3-2$”, which is represented as two sub-programs: “$+, 5, 3$” and “$-, \#0, 2$”. At the beginning, the Operator Generator produces “+” by sampling from the OP decoding vocabulary (refer to Equation (6)). Next, the generation for the operator is suspended, and the Reasoning Manager guides Operands Generator to produce “3” and “2” sampling from the NUM Decoding Vocabulary (refer to Equation (7)). After the first sub-program is complete, the Memory Register replaces the first cache token “#0” embedding with the guidance vector from Reasoning Manager. When generating the second sub-program, the Reasoning Manager enables the Operator Generator to produce the operator “$-$” for the second sub-program. Since the first operand in equation “$8 - 2$” refers to the executable result from the first sub-program. As a result, the Operands Generator produces “#0”, which refers to the executable result of the first sub-program. Finally, after operand “2” is generated, the generation for the second sub-program completes. Likewise, the Memory Register updates the second cache token “(#1)” embedding with the guidance vector.

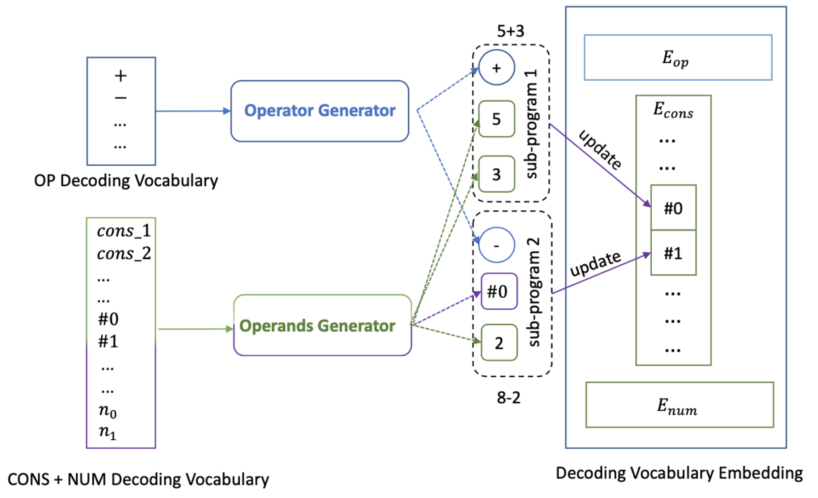


Figure A.2: An Example to Show How Separated Modules Work.

A.3 Comparison between Operation Length of the Golden Numerical Reasoning Program in MathQA and FinQA Datasets

Figure A.3 compares the distribution of operation length of the golden numerical reasoning program in FinQA and MathQA datasets. We can see that the lengths of operation of most numerical reasoning programs in FinQA are between 2 and 4. In contrast, MathQA contains more golden numerical reasoning programs with operation lengths between 3 and 8. Obviously, MathQA contains longer numerical reasoning programs than FinQA. As a result, the MathQA dataset is more complicated than the FinQA dataset.

A.4 Training (RoBERTa-large) on Extended FinQA Dataset)

One advantage of our ELASTIC model is that it is adaptable to the number of operands of an operator. We demonstrate this by evaluating ELASTIC on the MathQA dataset in the “Overall Results” section. However, another dataset we used for the evaluation,

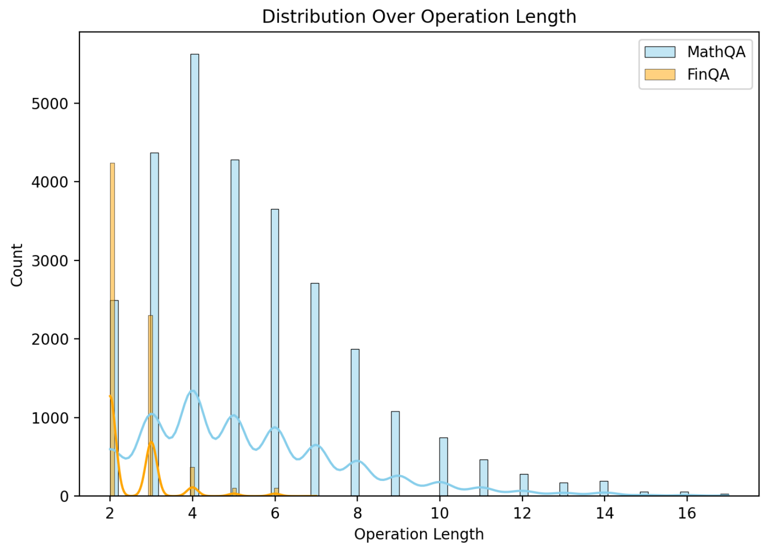


Figure A.3: The distribution of Operation Length in FinQA and MathQA datasets.

the FinQA dataset, only contains questions solved by operators with two operands. To test the advantage of our model on the FinQA dataset, we manually extend it by adding 30 and 20 questions for train and test data (named extended FinQA dataset), respectively (see Table A.2 for one example of the extended questions). These questions are proposed based on the original passages in the FinQA dataset. In addition, they are about superlative questions, which require to be solved by using superlative operators (i.e., *smallest* and *biggest*). As a result, unlike questions from the original FinQA dataset, the numbers of operands used to solve these extended questions are not limited to two. Next, We trained ELASTIC (RoBERTa-large) on the train data from the combination of the extended FinQA dataset and the original FinQA dataset. Since the number of operands of an operator is not determined anymore, the Reasoning Manager of ELASTIC has to manage the Operands Generator to generate the correct number of operands in terms of the specific question. This increases the difficulty for the model to generate correct operands and makes the dataset more challenging. The results are shown in Table A.1. For the performance on the combined test data (original FinQA + extended FinQA), ELASTIC (RoBERTa-large) achieves slightly lower scores (64.5 of Exec Acc and 63.8 of Prog Acc), compared to the results of ELASTIC (RoBERTa-large) achieved on original FinQA dataset (68.96 of Exec Acc and 65.21 of Prog Acc).[1] We also report the metric scores of ELASTIC (RoBERTa-large) achieved on test data from the extended FinQA dataset: 90.0 on both Exec Acc and Prog Acc. Note that the state-of-the-art model FinQANet cannot solve the extended FinQA dataset because it can only generate operators with two operands. These results show ELASTIC model solving questions that require the capability of generating operators with diverse numbers of operands.

Dataset (Test)	Exec Acc	Prog Acc
original FinQA + extended FinQA	64.5	63.8
extended FinQA	90.0	90.0

Table A.1: The performances of ELASTIC (RoBERTa-large) on the test data from the combination of the original FinQA dataset and extended FinQA dataset, and only on the test data from the extended FinQA dataset. Note that the model is trained on the train data from the combination of the original FinQA dataset and the extended FinQA dataset.

Question	What is the biggest obligations of payments between 2007 and 2010?
Passage†	Contractual obligations and commercial commitments the following table (in thousands): The operating lease obligations of payments due by fiscal year total is $4819; The operating lease obligations of payments due by fiscal year 2007 is $1703; The operating lease obligations of payments due by fiscal year 2008 is $1371; The operating lease obligations of payments due by fiscal year 2009 is $1035; The operating lease obligations of payments due by fiscal year 2010 is $710; The total obligations of payments due by fiscal year 2007 is $1903; The total obligations of payments due by fiscal year 2008 is $1571; The total obligations of payments due by fiscal year 2009 is $1235; The total obligations of payments due by fiscal year 2010 is $710.
Prediction	*biggest*(1903, 1571, 1235, 710)

Table A.2: An example from the extended FinQA dataset. The "Prediction" refers to the generated numerical reasoning program from ELASTIC (RoBERTa-large). †: The passage is from the FinQA dataset, which is reorganised for better readability.

Table A.2 shows one example from the extended FinQA dataset. To solve this question, the model needs to select numbers relevant to the obligations of payments between 2007 and 2019, compare them, and select the biggest one. We observe that

[1]See full results in "Overall Results" section.

ELASTIC (RoBERTa-large) correctly selects the relevant numbers and applies the *biggest* operator to them. Worth mentioning, there are 9 numbers (4819, 1703, 1371, 1035, 710, 1903, 1571, 1235, 710) relevant to the "*payments*" in the question. ELASTIC only selects part of these numbers which are relevant to the "*obligations of payments*" asked by the question. This demonstrates that ELASTIC understands the aim of the question and solves it by generating the correct numerical reasoning program.

Appendix B

GeoEval Dataset

B.1 Source datasets for GeoEval-2000 dataset

Table B.1 presents the the information of source datasets for GeoEval-2000 dataset. Meanwhile, Figure B.1 visualizes the proportional contributions of these source datasets to the GeoEval-2000 subset, showcasing the variety and scope of the geometry problems collected from each source. Finally, Figure B.2 illustrates the varied distribution of geometric shapes within the GeoEval-2000 subset, highlighting the diversity of geometry concepts represented in this collection.

Source Dataset	Diagram	Diagram Descriptions	Quantity
Geometry3K	✓	✓	3001
PGPS9K	✓	✓	9022
UniGeo	✓	✗	4998 †
GeoQA+	✓	✗	2518
GeometryQA	✗	✗	1398
MATH	✗	✗	1349 ‡
MathQA	✗	✗	2625 ‡

Table B.1: The information of source datasets for GeoEval-2000 dataset. The "†" symbol indicates that proving problems from the UniGeo dataset have been excluded. The "‡" sign specifies that the count only pertains to geometry problems within the dataset, focusing on problems directly relevant to the GeoEval-2000's scope.

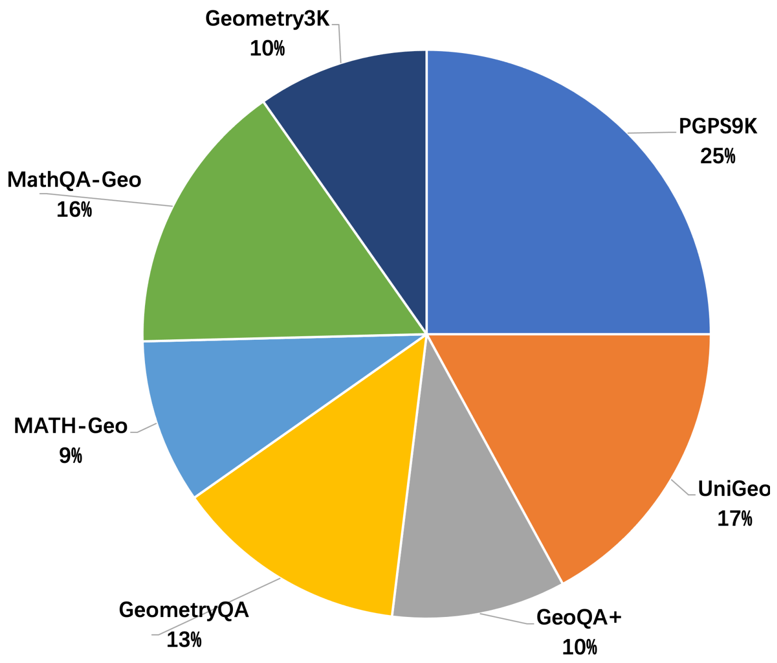


Figure B.1: Distributions of source datasets.

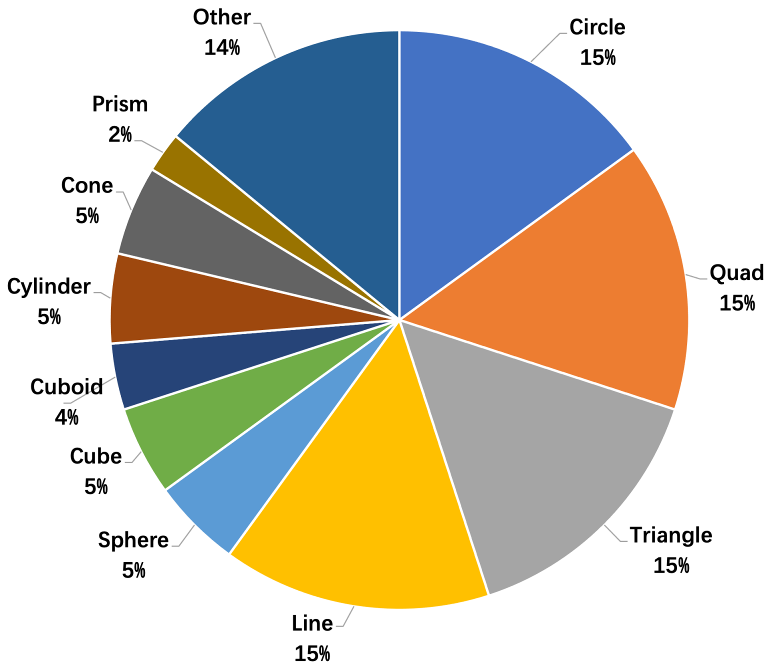


Figure B.2: Distributions of different geometric shapes.

B.2 Inspection of GeoEval-hard subset

To ensure the GeoEval-hard dataset's high quality and accuracy, we form a team of six reviewers, each holding at least a Master's degree, to scrutinize every question. This evaluation process is structured in three phases: individual review, swap review, and candidate review. The primary focus lies on two key standards: the completeness and relevance of the geometric diagrams, and the reasonableness of the answers provided.

In the first phase, "individual review", each reviewer is randomly assigned 50 geometry math problems from the GeoEval-hard dataset. Their task is to assess the geometry math problems based on the standards, marking any that fail to meet these standards. During the "swap review" phase, these sets of 50 geometry math problems are exchanged among reviewers for a second evaluation. To ensure unbiased assessment, we hide the results of the initial review. Here, reviewers again highlight geometry math problems not conforming to the standards. The final phase, "candidate review", involves selecting geometry math problems for the dataset based on the outcomes of the first two phases. Geometry math problems unmarked in both phases are retained, those marked in both are discarded, and those highlighted in only one phase undergo further examination by the entire review team, with the majority decision determining their inclusion.

B.3 Examples from GeoEval Representing Five Features

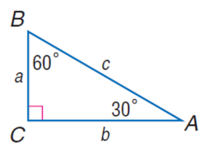
B.3.1 Comprehensive Variety

Figure B.3 present sample data from the GeoEval-2000 subset, illustrating its diversity in terms of data sources.

B.3.2 Varied Problems

Figure B.4 displays examples of three distinct problem types in the GeoEval benchmark: flat geometry, analytic geometry, and solid geometry.

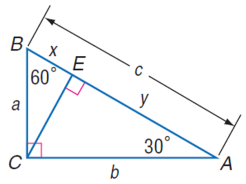
GeoEval – 2000 from PGPS9K
- Problems: If c = 5, find b.
- Diagram Descriptions:
  - structure: ["line B A", "line C A", "line B C"],
  - semantic: ["CA \\perp BC on C", "BA = c", "BC = a", "AC = b", "m \\angle ABC = 60", "m \\angle BAC = 30"]
- Choice List: A. 1.7, B. 2.6, C. 3.5, D. 4.3
- Solution Program: Equal c N0 Gsin N3 N1 N4 Get b



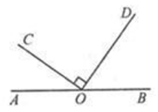
GeoEval – 2000 from MathQA
- Problems: eight cubes , each with a volume of 512 cm ^ 3 , are joined to form one large cube . what is the surface area of the large cube ?
- Diagram Descriptions: n/a
- Choice List: A. 4096 sq cm, B. 1536 sq cm, C. 1024 sq cm, D. 2048 sq cm, E. 512 sq cm
- Program: multiply(const_2,const_4)|multiply(n0,#0)|cube_edge_by_volume(#1)|surface_cube(#2)

GeoEval – 2000 from GeometryQA
- Problems: The length, width, and height of a rectangular box are (4/5) meter, (3/4) meter, and (1/6) meter, respectively. What is the maximum amount of water that can be held in this box?
- Diagram Descriptions: n/a
- Choice List: A. 1/10, B. 1/12, C. 1/15, D. 1/20
- Program: x=cuboid_volume((4/5), (3/4), (1/6))\nx=(4/5)*(3/4)*(1/6)

GeoEval – 2000 from Geometry3K
- Problems: If x = 7\\sqrt{3}, find b.
- Diagram Descriptions:
  - structure: ["line B C", "line C A", "line B E A", "line E C"],
  - "semantic": ["BA \\perp EC on E", "BC \\perp CA on C", "BC = a", "CA = b", "BE = x", "AE = y", "BA = c", "m \\angle EAC = 30", "m \\angle EBC = 60"]
- Choice List: A. 7.0, B. 12.1, C. 24.2, D. 42.0
- Program: Equal x N0 Gtan V0 N3 N7 Gsin V0 N2 N6 Get b



GeoEval – 2000 from UniGeo
- Problems: As shown in the figure, point O is on the straight line AB and OC ⊥ OD, if ∠COA = 36.0, then the size of ∠DOB is ()
- Diagram Descriptions: n/a
- Choice List: A. 36.0, B. 54.0, C. 64.0, D. 72.0
- Solution Program: g_minus C_3 N_0 g_minus V_0 C_2



GeoEval – 2000 from Math-Geometry
- Problems: Compute $\\sin 60^\\circ$.
- Diagram Descriptions: n/a
- Choice List: A. \frac{\sqrt{1}}{2}}, B. \frac{\sqrt{3}}{2}}, C. \frac{\sqrt{5}}{8}}, D. \frac{\sqrt{1}}{7}}
- Solution Program: n/a

GeoEval – 2000 from GeoQA+
- Problems: As shown in the diagram, the diagonals AC and BD of square ABCD intersect at point O. Point M is on side AD, and OM is connected. A perpendicular line is drawn from point O to OM, intersecting CD at point N. If the area of quadrilateral MOND is 2, then what is the length of BD?
- Diagram Descriptions: n/a
- Choice List: A. 1.4, B. 2.0, C. 2.8, D. 4.0
- Solution Program: g_double N_0

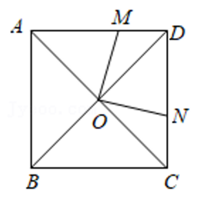
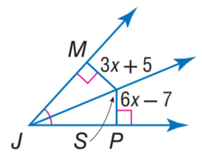


Figure B.3: Examples from GeoEval-2000 dataset. The golden answer choice is highlighted in red color.

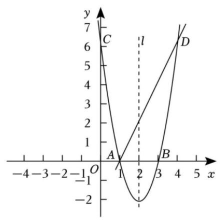
GeoEval – 2000 flat geometry

- Problems: If ST = 8, TR = 4, and PT = 6, find QR.
- Diagram Descriptions:
    - "structure": ["line Q R", "line S T R", "line Q P S", "line P T"],
    - "semantic": ["PT \\parallel QR"]
- Choice List: A. 6.0, B. 8.0, C. 9.0, D. 10.0
- Solution Program: g_double N_0



GeoEval – 2000 analytic geometry

- Problems: The graph shows a parabola $y=a x^{2}+b x+c \\quad(a, b, c$ are constants, and $a \\neq 0)$ passing through three points $A(1,0), B(3,0), C(0,6)$. Find the expression of the parabola.
- Diagram Descriptions: n/a
- Choice List: A. $y=2 x^{2}-8 x+6$, B. $y=-2 x^{2}+8 x+6$, C. $y=2 x^{2}+8 x+6$, D. $y=-2 x^{2}-8 x+6$
- Solution Program: n/a



GeoEval – 2000 solid geometry

- Problems: What is the volume of the triangular pyramid P-B1AM in the figure below? ABC-A1B1C1 is a right prism with angle ABC equal to 90 degrees, AB=BC=2, AA1=2, M is the midpoint of BC, N is the midpoint of A1C1, point P is on the line segment B1N, point Q is also on the line segment B1N, but first on the line segment AM, and AQ=2/3AM. S is the intersection of AC1 and A1C. If PS is parallel to the plane B1AM..
- Diagram Descriptions: n/a
- Choice List: A. 2/3, B. 1/3, C. 2/5, D. 3/7
- Solution Program: n/a

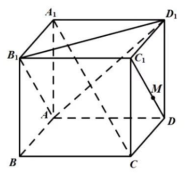


Figure B.4: Examples of the flat geometry problem, the analytic geometry problem, and the solid geometry problem in GeoEval benchmark.

B.3.3 Dual Inputs

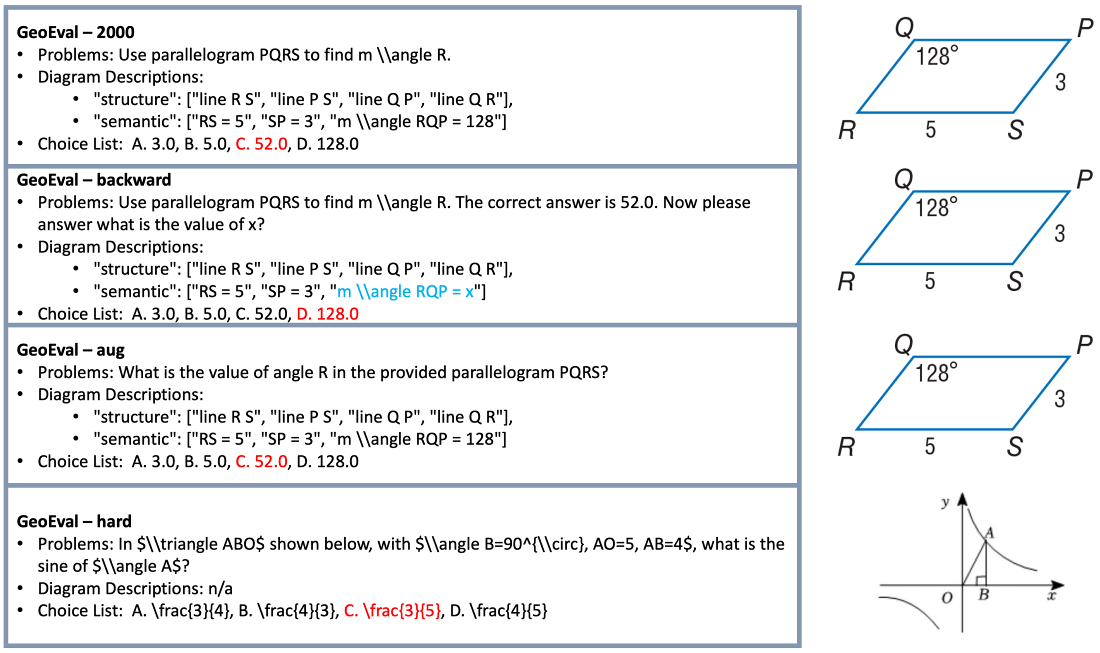
Figure B.3 shows that the GeoEval benchmark comprises geometry math problems that contain both diagrams and textual descriptions, as well as problems that include textual descriptions alone.

B.3.4 Diverse Challenges

Figure B.5 showcases examples from the GeoEval-2000, GeoEval-backward, GeoEval-aug, and GeoEval-hard subsets, illustrating the diverse challenges within the GeoEval benchmark.

B.3.5 Complexity Ratings

Every problem in the GeoEval benchmark is annotated with a complexity rating, indicating the level of skill necessary to solve it, as shown in Figure B.6.



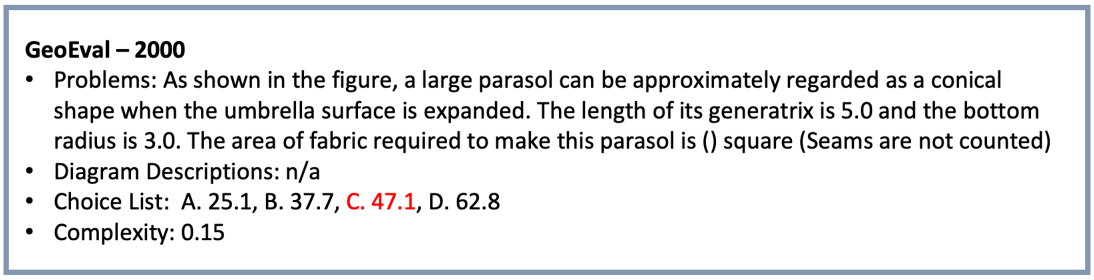
GeoEval – 2000
- Problems: Use parallelogram PQRS to find m \\angle R.
- Diagram Descriptions:
  - "structure": ["line R S", "line P S", "line Q P", "line Q R"],
  - "semantic": ["RS = 5", "SP = 3", "m \\angle RQP = 128"]
- Choice List: A. 3.0, B. 5.0, C. 52.0, D. 128.0

GeoEval – backward
- Problems: Use parallelogram PQRS to find m \\angle R. The correct answer is 52.0. Now please answer what is the value of x?
- Diagram Descriptions:
  - "structure": ["line R S", "line P S", "line Q P", "line Q R"],
  - "semantic": ["RS = 5", "SP = 3", "m \\angle RQP = x"]
- Choice List: A. 3.0, B. 5.0, C. 52.0, D. 128.0

GeoEval – aug
- Problems: What is the value of angle R in the provided parallelogram PQRS?
- Diagram Descriptions:
  - "structure": ["line R S", "line P S", "line Q P", "line Q R"],
  - "semantic": ["RS = 5", "SP = 3", "m \\angle RQP = 128"]
- Choice List: A. 3.0, B. 5.0, C. 52.0, D. 128.0

GeoEval – hard
- Problems: In $\\triangle ABO$ shown below, with $\\angle B=90^{\\circ}, AO=5, AB=4$, what is the sine of $\\angle A$?
- Diagram Descriptions: n/a
- Choice List: A. \frac{3}{4}, B. \frac{4}{3}, C. \frac{3}{5}, D. \frac{4}{5}

Figure B.5: Examples of GeoEval-2000, GeoEval-backward, GeoEval-aug, GeoEval-hard subsets.



GeoEval – 2000
- Problems: As shown in the figure, a large parasol can be approximately regarded as a conical shape when the umbrella surface is expanded. The length of its generatrix is 5.0 and the bottom radius is 3.0. The area of fabric required to make this parasol is () square (Seams are not counted)
- Diagram Descriptions: n/a
- Choice List: A. 25.1, B. 37.7, C. 47.1, D. 62.8
- Complexity: 0.15

Figure B.6: Example for a problem annotated with complexity in the GeoEval benchmark.

B.4 Algorithm for Classifying Geometry Math Problems Complexity

Algorithm 1 details our methodology for classifying each geometry math problem into distinct levels of complexity.

Algorithm 1: Algorithm for classifying geometry math problems complexity

```
Input: All Problem Texts $T$, All Diagram Descriptions $D$,
All Golden Solution Programs $S$
Output: Complexity for each problem
$len_{T,D} = 0$;
$len_S = 0$;
for $t$ in $T$, $d$ in $D$, $s$ in $S$ do
    $len_{T,D} += len_t + len_d$ ;   /* sum up the length of problem texts and
     the length of diagram descriptions.  */
    $len_S += len_s$ ;  /* sum up the length of golden solution programs.
     */
end
for $t$ in $T$, $d$ in $D$, $s$ in $S$ do
    $C_{t,d,s} \leftarrow \alpha \times \frac{len_t + len_d - \min(len_{T,D})}{\max(len_{T,D}) - \min(len_{T,D})} + (1 - \alpha) \times \frac{len_s - \min(len_S)}{\max(len_S) - \min(len_S)}$;
    if $0.0 \le C_{t,d,s} \le 0.2$ then
        Complexity ← Easy;   /* classify the problem as Easy problem.
         */
    else if $0.2 < C_{t,d,s} \le 0.6$ then
        Complexity ← Middle;          /* classify the problem as Middle
         problem.  */
    else if $0.6 < C_{t,d,s} \le 1.0$ then
        Complexity ← Hard;   /* classify the problem as Hard problem.
         */
    end
end
```

Appendix C

GAPS: Geometry-Aware Problem Solver

C.1 Hierarchical Beam Search

C.2 Accelerated Convergence with Problem-Type Classifier in GAPS Training

We have generated Figure C.1 to visualise the training losses of GAPS with and without the problem-type classifier. The plot illustrates that GAPS equipped with the problem-type classifier converges faster than GAPS without it. The reason behind this accelerated convergence lies in the problem-type classifier's role in simplifying the task of generating solution programs. By initially classifying the input geometry maths problem, the problem-type classifier provides valuable insights, streamlining the subsequent process of generating solution programs for the GAPS model.

C.3 Case Study for the Geometric Element Enhancement Technology

We also undertake a case study where we map out the probability distributions during the generation of operands, as seen in Figure C.2. This instance is taken from the proof

Algorithm 2: Hierarchical Beam Search

```
Input: $max_{op}, max_{oe}, bs, i, j = 0, op_{pred} =$ [“sos” for _ in range $bs$], $oe_{pred} = [\,[\,]\,]$
Output: $\mathcal{R}$
while $i \neq max_{op}$ do
    $ops, ops_{scores} \leftarrow$ Equation (6) ;    /* $ops$, $ops_{scores}$ are all predicted operands and probabilities, respectively. */
    $op_{candidates} \leftarrow$ select_max($ops, ops_{scores}, bs$) ;  /* select the predicted operators of the beam size $bs$ with the maximum probabilities. */
    $beam_{sub\_oe} = [\,]$;
    for $op$ in $op_{candidates}$ do
        $beam_{prev\_oe} = [\,[\,]$ for _ in range $max_{oe}$];
        while $j \neq max_{oe}$ do
            $oes, oes_{scores} \leftarrow$ Equation (6) ;    /* $oes$, $oes_{scores}$ are all predicted operands and probabilities, respectively. */
            $oe_{candidates} \leftarrow$ select_max($oes, oes_{scores}, bs$) ;    /* select the predicted operands of the beam size $bs$ with the maximum probabilities. */
            if $beam_{prev\_oe}[-1] \neq none$ then
                $prev_{oe\_ids} = oe_{candidates}//bs$ ;    /* get the previous operands ids of the selected operands */
                $beam_{prev\_oe} \leftarrow beam_{prev\_oe}[-1][prev_{oe\_ids}]$ ; /* save the previous operands of the selected operands */
            end
            else
                $beam_{prev\_oe} \leftarrow oe_{candidates}$
            end
            j +=1;
        end
        $beam_{sub\_oe} \leftarrow$ backtrace($beam_{prev\_oe}[-1]$) ;    /* backtrace to recover the selected operands sequences */
        $beam_{sub\_oe} \leftarrow oe_{candidates}$ ;    /* remember to append the last step prediction to the sequences */
    end
    $score_{op\_oe} \leftarrow$ merge_score($op_{candidates}, beam_{sub\_oe}$) ;  /* merge_score: sum the probabilities of operators and their operands */
    $candidates_{op\_oe} \leftarrow$ select_max($score_{op\_oe}, bs$) ;  /* selects beam size bs of the combinations of operator and operands with highest probabilities. */
    $prev_{op\_ids} \leftarrow candidates_{op\_oe}//bs$ ;    /* get the previous operators ids of the selected sub-program */
    $op_{pred} \leftarrow op_{pred}[-1][candidates_{op\_oe}]$ ;    /* save the previous operators of the selected sub-program */
    $oe_{pred} \leftarrow beam_{sub\_oe}[prev_{op\_ids}]$;
    i +=1;
end
$\{op\}, \{op_{index}\} \leftarrow$ backtrace($op_{pred}$) ; /* backtrace to recover the selected operator sequences */
$\{oe\} \leftarrow oe_{pred}[\{op_{index}\}]$ ;    /* get the operands of the selected operator sequences */
$\{op\} \leftarrow candidates_{op\_oe}.op$ ;  /* remember to append the last step prediction to the sequences */
$\{oe\} \leftarrow candidates_{op\_oe}.oe$;
return $\mathcal{R} = \{op\} \vee \{oe\}$
```

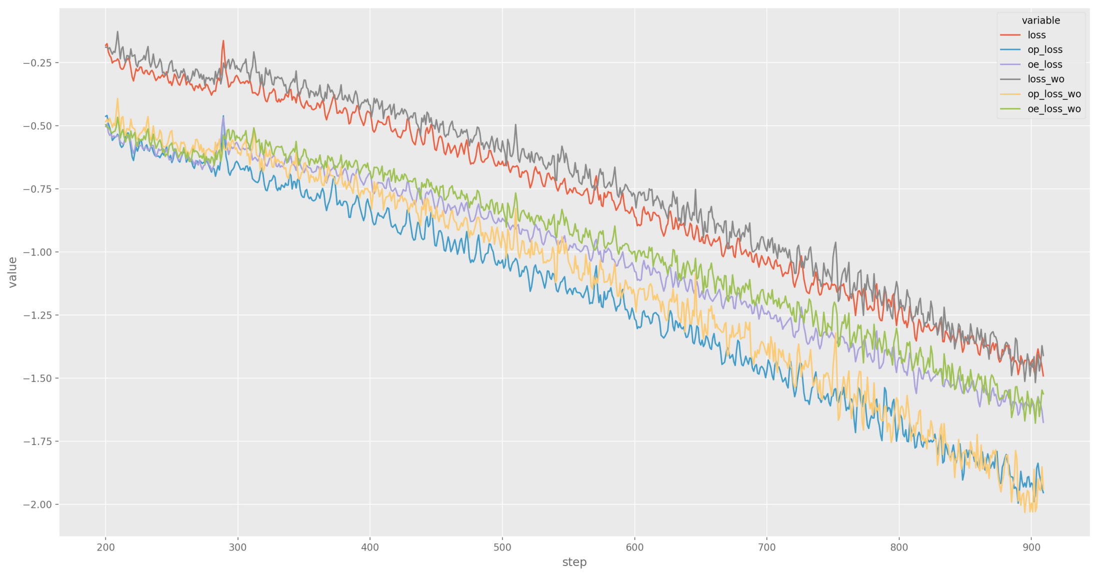


Figure C.1: Comparison of training loss convergence in GAPS, with and without the utilisation of the problem-type classifier. The "loss" and "loss_wo" are the sum of operator and operands training losses values with and without problem-type classifier. The "op_loss" and "op_loss_wo" are the operator training loss values with and without problem-type classifier. The "oe_loss" and "oe_loss_wo" are the operand training loss values with and without problem-type classifier. The loss values are scaled by the logarithmic function for better visualisation.

problem showcased in Figure 1.1, where the intended solution program is designated as "R_4, E_1, congruent, E_3, R_15, E_3, similar, E_2". Observing Figure C.2, it is evident that GAPS is inclined to choose geometric elements from the appended geometric elements while employing geometric element enhancement. Conversely, in the absence of geometric element enhancement, GAPS erroneously picks "E_0" ("triangle SUW") as the operand for the second sub-program, whereas the accurate operand should be "E_3" ("angle SUW"). We hypothesise this occurs due to the geometric elements being tokenized into multiple segments by the tokenizer, allowing these segments to easily engage with other contextual token pieces, subsequently generating extra noise. Nonetheless, appending these geometric elements to the problem statement can alleviate such noise, facilitating an enhancement in the precision of operand selection.

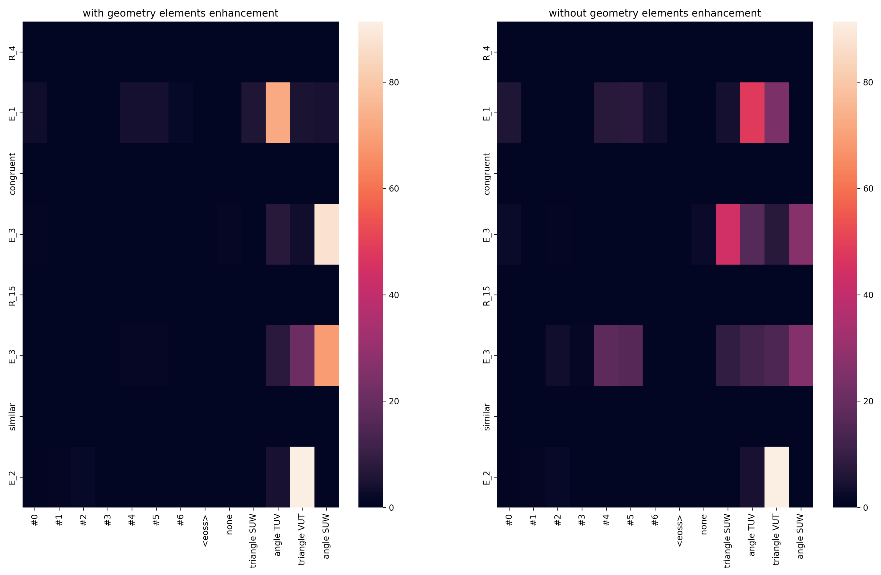


Figure C.2: This heatmap illustrates the probability distributions across operand generation with and without the utilisation of geometric element enhancement technology. It exemplifies the creation of a solution program for a geometry proving problem. We truncate the x-axis by omitting the constants not pertinent to solving proving problems. The y-axis denotes the tokens in the solution program, and the probabilities of generating operators are nullified, as this instance aims to scrutinise the impact of geometric element enhancement on the generation of operands. With the enhancement enabled, the value vectors for the geometric elements are derived from the vectors of the corresponding appended geometric elements in the problem text. Conversely, without the enhancement, the value vectors for the geometric elements are extracted from the vectors of the original geometric elements within the problem text.

Appendix D

GOLD: Geometry Problem Solvers with Natural Language Description

D.1 Preliminary: FCOS and GSM Models

This section equips the reader with the foundational knowledge required for comprehending the Section 6.2. Particularly, we delve into the FCOS model [8], which plays a crucial role in detecting the symbols outlined in Section 6.2.1. Additionally, we explore the GSM model [183], an instance segmentation method integral for extracting geometry primitives as defined previously.

D.1.1 FCOS: Object Detection

FCOS model [8] is a type of deep learning model used primarily for object detection tasks. It represents a shift from the traditional two-stage detection frameworks (like Faster R-CNN [192]) to the one-stage approach. Figure D.1 displays the overall architecture of the FCOS network.

Unlike traditional methods that rely on region proposal networks (RPNs) to generate candidate object locations, FCOS falls into the category of one-stage detectors, like YOLO [239] and SSD [240]. Therefore, FCOS is generally faster than two-stage

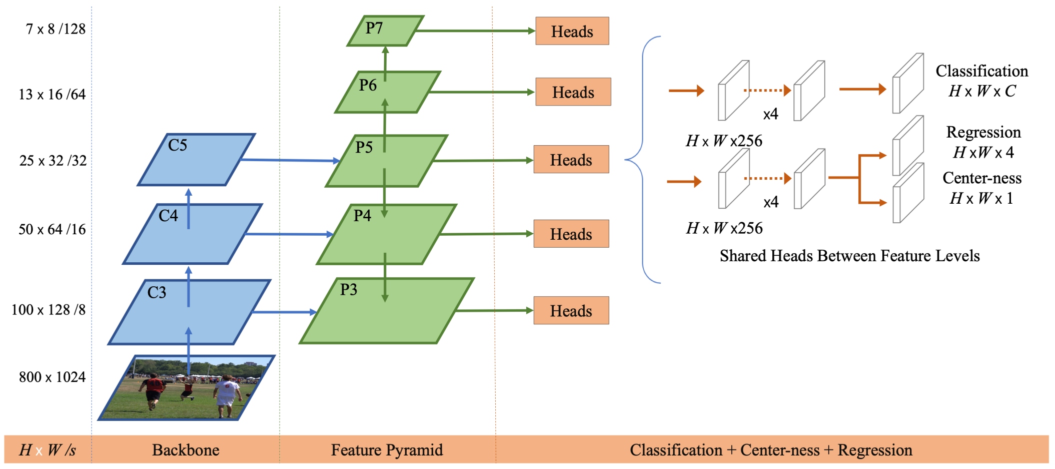


Figure D.1: The illustration of the FCOS network. The C2, C3, C4, C5 refer to the feature maps from the backbone network, such as ResNet-101 [6] and MobileNets [7]. Each of them are obtained by using different ratio (denotes as $/s$ (s = 8, 16, 32, 64, 128) to down-sample over the input image with the size of 800×1024 in this example. The figure is taken from the FCOS paper [8].

detectors as it does not have a separate region proposal step. They directly predict the bounding boxes and class probabilities in a single pass. Additionally, traditional object detectors use anchor boxes – predefined boxes of various sizes and aspect ratios – to detect objects. FCOS, however, is anchor-free. It does not rely on these pre-defined anchor boxes, which simplifies the training process.

To obtain the bounding box of the object, FCOS model adopts three tasks: bounding box regression, classification, and the centerness prediction.

D.1.1.1 Bounding Box Regression

In FCOS model, each location x, y on the feature maps is considered as a potential object centre. For each location, the model predicts four offsets l, t, r, b which represent the distances from this location to the left, top, right, and bottom sides of the bounding box, respectively.

If x, y is the location on the i-th feature map and x_a, y_a is the corresponding location on the original input image:

$$
\begin{aligned}
x_a &= x \times s + \left\lfloor \frac{s}{2} \right\rfloor \\
y_a &= y \times s + \left\lfloor \frac{s}{2} \right\rfloor
\end{aligned}
\tag{D.1}
$$

where s is the scaling factor of the i-th feature map.

Next, the predicted bounding box B_x, B_y, B_w, B_h in the original image can be calculated as:

$$
\begin{aligned}
B_x &= x_a - l \times s \\
B_y &= y_a - t \times s \\
B_w &= l + r \times s \\
B_h &= t + b \times s
\end{aligned}
\tag{D.2}
$$

where B_x and B_y refer to the coordinates of the top-left corner of the bounding box in the original image, and B_w and B_h are the width and height of the bounding box.

D.1.1.2 Object Classification

For classification, the model predicts a probability distribution over C classes for each location x, y. If p_c is the predicted probability for class c, the classification loss (or named focal loss) for a positive sample can be:

$$
\mathrm{L}_{\text{focal}} = \alpha(1 - p_c)^{\gamma} log(p_c) \tag{D.3}
$$

where α and γ are the hyperparameters of the focal loss, designed to mitigate the problem of class imbalance by down-weighting the loss assigned to well-classified examples.

D.1.1.3 Centerness Prediction

During inference, there might be a number of points be regarded as the class c. FCOS tends to select point which is close to the real centre point. Therefore, FCOS predict a scalar value $s \in [0, 1]$ that gauges how close location is to the centre of an object. It's calculated as:

$$s = \sqrt{\frac{min(l,r)}{max(l,r)} \cdot \frac{min(t,b)}{max(t,b)}} \tag{D.4}$$

when s equals to 1, the predicted point is at the centre point, whereas when s equals to 0, the predicted point is the margin point.

D.1.1.4 Train the FCOS Network

The overall loss function for FCOS is a combination of the classification loss, the box regression loss, and the centerness loss. The box regression loss is typically a form of IoU loss or smooth L1 loss. The total loss for a positive sample is:

$$\mathrm{L} = \mathrm{L}_{\text{focal}} + \lambda_1 \mathrm{L}_{box} + \lambda_2 \mathrm{L}_{\text{centerness}} \tag{D.5}$$

where λ_1 and λ_2 are hyperparameters balancing the different components of the loss.

D.1.2 GSM: Instance Segmentation

The Geometric Segmentation Module (GSM), as detailed in [183], is an instance segmentation method designed for the extraction of geometric primitives from diagrams. The architecture of GSM comprises two primary branches: a semantic segmentation branch and a segmentation embedding branch. The core objective of the GSM model is to optimize performance by minimizing the cross-entropy loss for each pixel in the original image:

$$\mathrm{L}_{bs*} = \frac{-w_*}{M_{map}} \sum_{i=1}^{M_{map}} y_i^* log(p_i^*) + (1 - y_i^*) log(1 - (p_i^*) \tag{D.6}$$

where $*$ denotes that the pixel belongs to one of primitive classes, w is for balancing the positive and negative classes. And M_{map} is the total pixel number.

To better differentiate pixels of one primitive from pixels of other primitives, GSM add regularisation terms [241] to the Equation D.6:

$$
\begin{aligned}
\mathrm{L}_{dist} &= \frac{1}{N(N-1)} \sum_{n_1=1}^{N} \sum_{n_2=1, n_2 \neq n_1} [2\delta_d - ||\mu_{n1} - \mu_{n2}||]_+^2 \\
\mathrm{L}_{var} &= \frac{1}{N} \sum_{n_1=1}^{N} \frac{1}{M_n} \sum_{i=1}^{M_n} [||\mu_n - x_i|| - \delta_v]_+^2
\end{aligned}
\tag{D.7}
$$

where N is the sum of the number of pixels belonging to primitives. x is the pixel embedding, μ is the centre of instance embedding. Intuitively, L_{dist} tries to increasing the distance between the cluster centres of different ptimitives. Conversely, L_{var} tends to push embeddings of pixels of the same primitive to the centre of the cluster.

D.2 Convert Relations to Natural Language Descriptions

Relations		Paradigm	Example
geo2geo	Point	The diagram contains ${}.	The diagram contains **Point A**, **B**, **C**.
	Line	The diagram contains ${}, which has endpoints: ${} and ${}, In addition, there is/are ${} on the line.	The diagram contains **Line L1**, which has endpoints: **Point P0** and **Point P1**, In addition, there is/are **Point P2** on the line.
	Circle	The diagram contains ${}, whose center point is ${}, which has ${} on its arc.	The diagram contains **Circle M**, whose center point is **Point E**, which has **Point F**, **Point G** on its arc.
text2geo	Degree	1. Angle ${} has degree of ${}. 2. Line ${} and Line ${} cross at Point ${} has degree of ${}.	1. Angle **1** has degree of **100**. 2. Line **L1** and Line **L2** cross at Point **C** has degree of **50**.
	Length	The length of Line ${} between Point ${} and Point $ is ${}.	The length of Line **L3** between Point **A** and Point **B** is **10**.
	Circle Degree	Line ${} and Line ${} cross at the center point ${} of Circle ${} has degree of ${}.	Line **L1** and Line **L2** cross at the center point **C** of Circle **C0** has degree of **20**.
other2geo	same degree	Angle ${} has the same degree with Angle ${} ...	Angle **1** has the same degree with Angle **2**, Angle **3**.
	same length	Line ${} has the same length with Line ${} ...	Line **L1** has the same length with Line **L2**, Line **L3**.
	parallel	Line ${} is parallel with Line ${}...	Line **a** is parallel with Line **b**.
	perpendicular	Line ${} is perpendicular with Line ${} at Point ${}.	Line **L1** is perpendicular with Line **L2** at Point **C**.

Table D.1: The defined paradigm used to convert *geo2geo* and *sym2geo* relations to natural language descriptions $\mathcal{L}$. "${}" is the placeholder. The placeholder is filled in as demonstrated in the "Example" column, and the filled content is highlighted in bold type.

Once the *geo2geo* relations and *sym2geo* relations have been established, we proceed to convert these relations into natural language descriptions denoted as $\mathcal{L}$ following the guidelines specified in Table D.1.

To begin, we initiate the process by representing the existing geometric primitives in the diagram by enumerating points, lines, and circles within the description of the *geo2geo* relation. In detail, we sequentially enumerate all existing points, providing their reference names as described in the "Point" entry of Table D.1. We describe the associated points for each line by mentioning their reference names. Additionally, we include a list of points that have "end-point" and "on-a-line" relations with the line, as specified in the "Line" entry of Table D.1. Similarly, for each circle, we mention its reference name and proceed to list the points that exhibit "centre-point" and "on-a-circle" relations with the circle, following the guidelines provided in the "Circle" entry of Table D.1.

Next, we proceed to describe the *text2geo* relation within the *sym2geo* relation based on the predicted *text_class*. Here are the guidelines for each case:

- If the *text_class* indicates that the symbol refers to the reference name of a point (or a line, or a circle), we modify the name of the corresponding point (or line, or circle) accordingly.
- If the *text_class* indicates that the symbol refers to the degree of an angle, we describe it following the guidelines specified in the "Degree" entry of Table D.1.
- If the *text_class* indicates that the symbol refers to the length of a line, we describe it according to the instructions provided in the "Length" entry of Table D.1.
- If the *text_class* indicates that the symbol refers to the degree of an angle on the circle, we describe it based on the guidelines outlined in the "Circle Degree" entry of Table D.1.

Furthermore, when dealing with the *other2geo* relations, we describe them based on the specific type of geometric relation as indicated in Table D.1.

D.3 Instruction Choice

Instructions serve as direct and explicit commands that clearly communicate to the model the specific task it is required to perform. For our experiments, we initially selected two distinct instruction templates for Llama2-13b-chat [75] and CodeLlama-13b [225], as detailed in Table D.2. Upon experimental evaluation, it was observed that the instruction template modified from the one used to train the Llama2 model (displayed at the upper row in Table D.2) demonstrated superior performance. Consequently, we opted for this template.

IT	You are a problem-solving bot, and now I ask you to solve a geometry problem, please answer the question and provide the correct option letter. The problem is as follows: {Problem Text} Here are the basic descriptions of the diagram: {Natural Language Descriptions} The Answer and the Reason Process are: [/INST]	Hint: Please answer the question and provide the correct option letter, e.g., A, B, C, D, at the end {Problem Text} Here are the basic descriptions of the diagram: {Natural Language Descriptions}
EX	[INST] You are a problem-solving bot, and now I ask you to solve a geometry problem, please answer the question and provide the correct option letter. The problem is as follows: Find the perimeter of the polygon. The Choices are: A: 20.0, B: 24.0, C: 28.0, D: 34.409, Here are the basic description of the diagram: The diagram contains Point P0, Point P1, Point P2, Point P3, Point P4, The diagram contains Line L0, which has endpoints: Point P1, Point P3, Line L1, which has endpoints: Point P1, Point P4, Line L2, which has endpoints: Point P3, Point P4, Line L3, which has endpoints: Point P0, Point P3, Line L4, which has endpoints: Point P0, Point P1, Line L5, which has endpoints: Point P0, Point P4, The length of Line L0 between Point P2 and Point P3 is 7. The length of Line L4 between Point P2 and Point P1 is 7. The length of Line L5 between Point P4 and Point P2 is 5. Line L3 between Point P0 and Point P3 has the same length with Line L4 between Point P1 and Point P0 and Line L2 between Point P3 and Point P4 and Line L1 between Point P1 and Point P4. The Answer and the Reason Process are: [/INST]	Hint: Please answer the question and provide the correct option letter, e.g., A, B, C, D, at the end Here are the basic description of the diagram: The diagram contains Point P0, Point P1, Point P2, Point P3, Point P4, The diagram contains Line L0, which has endpoints: Point P1, Point P3, Line L1, which has endpoints: Point P1, Point P4, Line L2, which has endpoints: Point P3, Point P4, Line L3, which has endpoints: Point P0, Point P3, Line L4, which has endpoints: Point P0, Point P1, Line L5, which has endpoints: Point P0, Point P4, The length of Line L0 between Point P2 and Point P3 is 7. The length of Line L4 between Point P2 and Point P1 is 7. The length of Line L5 between Point P4 and Point P2 is 5. Line L3 between Point P0 and Point P3 has the same length with Line L4 between Point P1 and Point P0 and Line L2 between Point P3 and Point P4 and Line L1 between Point P1 and Point P4.

Table D.2: Two instruction templates. "IT" and "EX" refer to the instruction template and examples. The template in the left column is modified from the instruction used to train the Llama2 model, and another one is from the [12]. In the column of "IT", the "{problem Text}" is the geometry maths problem text $\mathcal{T}$, and "{Natural Language Descriptions}" is the description of the diagram $\mathcal{L}$.

Appendix E

Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving

E.1 Model Hyperparameters

Table E.1 presents the complete list of hyperparameters applied to the models throughout the evaluation phase.

Model Name	Generation Parameters	Comments
CodeGen2-16B	do_sample=True, top_k=0.5, top_p=0.5, max_tokens=512	model=""Salesforce/codegen2-16B"
WizardMath-7B-V1.1	temperature=0.0, top_p=1, max_tokens=1024	vLLM package
WizardMath-70B	temperature=0.0, top_p=1, max_tokens=1024	vLLM package
GPT-3.5	temperature=0.7, max_tokens=512	version="gpt-3.5-turbo-0125"
GPT-4	temperature=0.7, max_tokens=512	version="gpt-4-1106-preview"
llava-7B-V1.5	temperature=0.0, max_new_tokens=512	llava package
Qwen-VL	temperature=0.0, max_new_tokens=512	model="Qwen/Qwen-VL"
mPLUG-Owl2	do_sample=True, top_p=0.7, max_tokens=512	model="mPLUG-Owl2"
GPT-4V	temperature=0.0, max_tokens=512	version="gpt-4-vision-preview"

Table E.1: The hyperparameters for the models used in the evaluation are detailed. When the "comments" section includes the format *model = ""*, it signifies that the model was loaded from the transformer package. The vLLM package indicates that models are implemented by the vLLM package, where more details can be found in `https://github.com/vllm-project/vllm`. For models other than OpenAI's GPT, custom codes were utilized for evaluation unless specified otherwise in the comments.

E.2 Instruction Prompt Used for Evaluating Models

	Template	Example
Merge	Here are the basic description of the diagram: ${diagram descriptions}, ${problems texts}, The Choices are: ${choice list}	Please solve this math problem: Here are the basic description of the diagram: line B A, line C A, line B C\nCA \\perp BC on C, BA = c, BC = a, AC = b, m \\angle ABC = 60, m \\angle BAC = 30\nIf c = 5, find b. The Choices are: [1.7, 2.6, 3.5, 4.3]
Instruction	Please solve this math problem: ${Merge} ### Problem-solving Bot:	Please solve this math problem: Here are the basic description of the diagram: line B A, line C A, line B C\nCA \\perp BC on C, BA = c, BC = a, AC = b, m \\angle ABC = 60, m \\angle BAC = 30\nIf c = 5, find b. The Choices are: [1.7, 2.6, 3.5, 4.3] ### Problem-solving Bot:

Table E.2: Templates and examples provided illustrate the process of merging and instruction creation. The placeholder "${Merge}" represents the combined texts of "diagram descriptions," "problems texts", and "choice list". In cases where "diagram descriptions" are absent, the phrase "Here are the basic description of the diagram:" is omitted.

Prior to employing instruction prompts to steer model responses, we combine the problem texts, diagram descriptions, and choice lists from an example, as depicted in the "Merge" row of Table E.2. Following this combination, as illustrated in the "Instruction" row of Table E.2, we incorporate instruction prompts into the merged texts and then forward these to the models to generate responses.

E.3 Prompt and Heuristic rules For Answer Extraction

We detail the prompts utilized for extraction using GPT-4, which include an extraction instruction alongside various sample prompts. The extraction instruction and the constructed samples are presented in Table E.3 and Table E.4, illustrating the methodology behind the extraction process.

Element	**Prompt**
Task description	You are a result extraction bot. I will provide you with geometry questions and a model output, and you will help me extract the reference answers from the model's output.
Example 1	**Question:** As shown in the figure, in triangle ABC, $AB = AC$, $\angle A = 40°$, DE is the perpendicular bisector of AB. What is the degree measure of $\angle DBC$? () **Choices:** (A) 30.0 (B) 20.0 (C) 15.0 (D) 10.0 **Model response:** Given that $\angle A = 40°$, $AB = AC$, we can conclude that $\angle ABC = \angle ACB = 70°$. Also, since DE is perpendicular and bisects AB, we have $DB = AD$ and $\angle ABD = \angle A = 40°$. Therefore, $\angle DBC = \angle ABC - \angle ABD = 70° - 40° = 30°$. The answer is option A. **Extracted answer: A**
Example 2	**Question:** As shown in the diagram, the lateral surface area of a cone with a radius of the base r equals 6 cm and a height h equals 8 cm is (). **Choices:** (A) 94.2 (B) 150.8 (C) 188.5 (D) 251.3 **Model response:** Given $h = 8\,\text{cm}$ and $r = 6\,\text{cm}$, the length of the generatrix of the cone can be set as l. Using the Pythagorean theorem, we have $l = \sqrt{6^2 + 8^2} = 10\,\text{cm}$. The lateral area of the unfolded cone is $S_{\text{side}} = 0.5 \times 2 \times 6\pi \times 10 = 60\pi\,\text{cm}^2$. Therefore, the lateral area of the cone is $60\pi\,\text{cm}^2$. Therefore, the answer is C. **Extracted answer: C**
Example 3	**Question:** In triangle ABC, F is the midpoint of BC and point E is on the AC side. AC = 10. What is the length of AE? **Choices:** (A) 3.0 (B) 4.0 (C) 5.0 (D) 4.5 **Model response:** Since F is the midpoint of BC, EF is parallel to AB, so EF is the median of triangle ABC. Therefore, point E is the midpoint of AC. Therefore, AE = 0.5 × AC. Since AC = 10, AE = 5. Therefore, the answer is C. **Extracted answer: C**

Table E.3: Task specific instruction used for extracting the answer, and three examples.

Regular expressions	**Demonstration Examples**
`value of (\w+) is\s*([\d.]+)`	The value of x is 3.5.
`correct answer is\s*(.+).`	correct answer is C."
`answer is\s*([\d.]+)`	answer is 17.1."
`answer should be\s*(.+) degrees`	Therefore, the answer should be choice D."
`answer to (.+) is (.+) degrees`	The answer to the angle ABC is 60°
`answer to the problem is\s*(.+)`	The correct answer to problem is $y = x^2 + 2x + 3$."
`The closest (.+) is (.+).`	So we got the area is 13.1. The closest answer is D."
`the (.+) is equal to (.+).`	The degree measure of angle ABC is 35 degrees.
`(.+) is approximately (.+) units`	So, the length of the line segment is approximately 10 units."

Table E.4: Regular expressions used for extracting the answers that GPT-4 cannot tackle.

E.4 Results Across Different Subjects

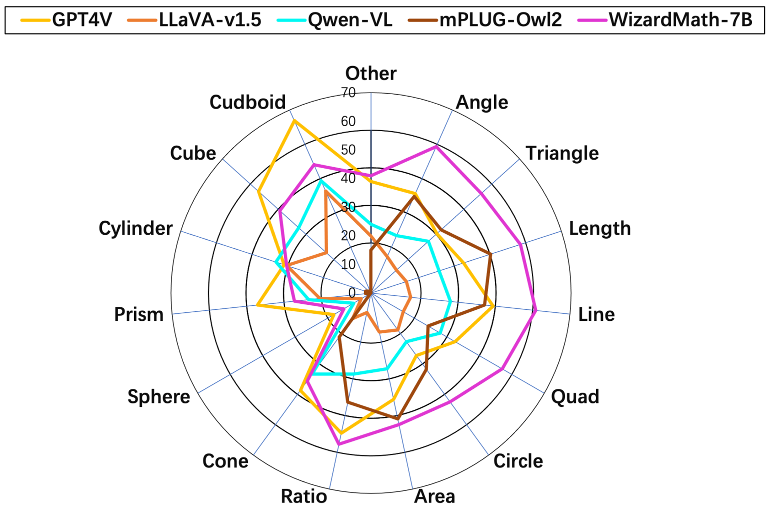


Figure E.1: Detailed accuracy scores for models across various academic subjects.

Figure E.1 displays the performance of models across various subjects, revealing distinct strengths. The WizardMath-7B model significantly outperforms others in flat geometry problems, such as length and lines. Conversely, in solid geometry problems like cuboids and spheres, GPT-4V surpasses WizardMath-7B, indicating its superior capability in addressing solid geometry questions.