

FGeo-HyperGNet: Geometry Problem Solving Integrating Formal Symbolic System and Hypergraph Neural Network

Xiaokai Zhang¹, Na Zhu^{1,2}, Yiming He^{1,2}, Jia Zou^{1,2}, Cheng Qin^{1,2}, Yang Li¹,
Zhenbing Zeng³, and Tuo Leng^{1,2}(✉)

¹ School of Computer Engineering and Science, Shanghai University, Shanghai, China
tleng@shu.edu.cn

² Institute of Artificial Intelligence, Shanghai University, Shanghai, China

³ College of Sciences, Shanghai University, Shanghai, China

Abstract. Geometry problem solving has always been a long-standing challenge in the fields of automated reasoning and artificial intelligence. This is the fifth article in a series of our works, we built a neural-symbolic system to automatically perform human-like geometric deductive reasoning. The symbolic part is a formal system built on FormalGeo, which can automatically perform geometric relational reasoning and algebraic calculations and organize the solving process into a solution hypertree with conditions as hypernodes and theorems as hyperedges. The neural part, called HyperGNet, is a hypergraph neural network based on the attention mechanism, including an encoder to effectively encode the structural and semantic information of the hypertree, and a solver to provide problem-solving guidance. The neural part predicts theorems according to the hypertree, and the symbolic part applies theorems and updates the hypertree, thus forming a predict-apply cycle to ultimately achieve readable and traceable automatic solving of geometric problems. Experiments demonstrate the correctness and effectiveness of this neural-symbolic architecture. We achieved a step-wised accuracy of 87.65% and an overall accuracy of 85.53% on the formalgeo7k datasets. The code and data is available at [here](#).

Keywords: Geometry problem solving · Neural-symbolic system · Hypergraph neural network

1 Introduction

Geometry problem solving (GPS) has always been a long-standing challenge [9,32,17] in the fields of mathematical reasoning and artificial intelligence, owing to the cross-modal forms of knowledge and the absence of automated solving methods. As depicted in Fig. 1, GPS can be described as: given a description (original images and texts or formalized) of a geometric problem, the solver needs to implement step-wised reasoning leading to the final answer.

Traditional methods of GPS can generally be divided into three categories. The first category is the synthesis methods, such as backward search method [11]

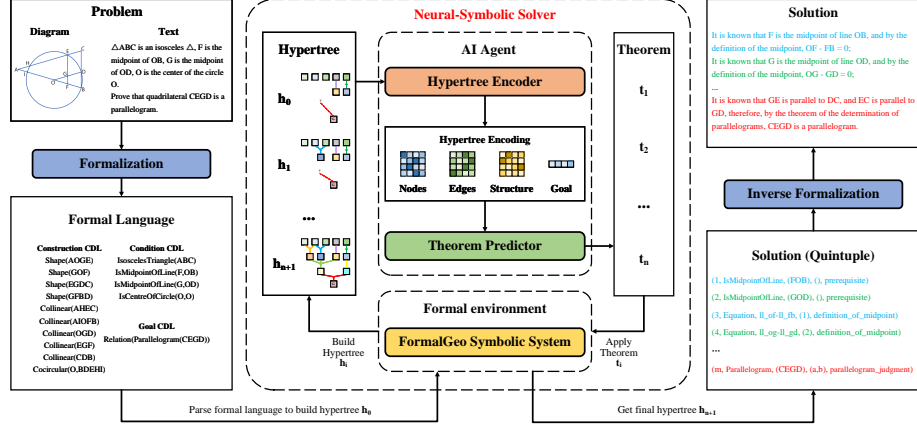


Fig. 1. The overall architecture of proposed neural-symbolic system.

, forward chaining method [19] and deductive database approach [8]; the second category is the algebraic methods based on coordinates, such as Wu’s method [30] and Gröbner bases method [2]; the third category is the point elimination methods based on geometric invariants [37], which can be extended to solid geometry [7] and non-Euclidean geometry [34].

Artificial intelligence technology has provided new ideas for GPS [25,24,10]. In particular, with the rapid development and application of deep learning and Large Language Models, a series of neural-symbolic methods have been proposed. These methods can generally be divided into two categories [21]: symbolic approaches and probabilistic approaches, as shown in Tab. 1. The symbolic approaches [18] parse the images and texts of geometric problems into a unified formal language description, and then apply a predefined set of theorems to solve the problems. These approaches require the establishment of a formal system and the problem-solving process has mathematical rigor and good readability. The probabilistic approaches [6] view GPS as a sequence generation task with multi-modal input. These approaches learn from problem-solving examples to map geometric problem descriptions into interpretable programs. After program sequences are generated, the executor computes them step by step and obtain the solved answer.

Nevertheless, existing research has notable limitations. The majority of recent advancements in GPS primarily concentrate on exploring new learning methods and models [15,31,26], yet they overlook the investigation of geometric formal systems. The theorems and predicates of these formal systems are roughly implemented using programming languages, and the definition of new predicates and theorems necessitates modifications to the solver’s code. This characteristic significantly hampers the scalability of formal systems to problems of Olympiad-level [27]. Existing symbolic approaches to problem-solving are non-

Table 1. An overview of AI-assisted GPS methods.

Methods	Categories	Acc (%)				
		Geometry3K	GeoQA	GeoQA+	UniGeo	PGPS9K
Inter-GPS [18]	symbolic	78.30				68.00
NGS [6]	probabilistic	76.20	60.00	63.31	51.90	46.10
DPE-NGS [3]	probabilistic		62.65	66.09		
Geoformer [5]	probabilistic	59.30	60.30		62.50	47.30
GCN-based [12]	symbolic	74.90				
CL-based [15]	probabilistic		61.80			
PGPSNet [39]	probabilistic	77.90				70.40
SCA [20]	probabilistic	76.70	64.10			
GeoDRL [21]	symbolic	89.40				
LANS [38]	probabilistic	82.30				74.00
CFER [26]	symbolic	59.50				

* The data is sourced from the aforementioned references.

traceable, and the redundant theorems applied during heuristic searches cannot be eliminated. Existing probabilistic approaches, on the other hand, fail to yield a human-like problem-solving process, suffer from low readability, and cannot guarantee the correctness of results. The process of solving geometric problems encompasses both numerical computation and relational reasoning, with existing research predominantly focused on numerical problem-solving objectives, struggling to integrate computation and reasoning within a unified framework [5]. This imperfection in the existing formal systems severely restricts the types and complexity of geometric problems that can be solved.

In addition, existing work predominantly focuses on the unified cross-modal integration of geometric text and image [39,38] with limited attention to the embedding of geometric formal languages. The majority of studies treat geometric text as natural language sequences, overlooking the rich structural information inherent in geometric conditions. Neglecting of graph structure information in formal language results in poor theorem prediction [12]. S2G [28] maps the problem-solving process onto an expression tree, implicitly incorporating process information, yet it does not reflect a human-like problem-solving approach. GeoDRL [21] organizes geometric conditions into a Geometric Logic Graph (GLG), but the GLG lacks information about the problem-solving process and fails to model the interrelations among theorems. Formal language, distinct from natural language, adheres to stringent syntactic forms. Its symbols bear specific meanings and inappropriate tokenization can obliterate the inherent meaning of statements [20]. Moreover, In the field of natural language processing, sentence inputs can be represented as two-dimensional real number matrices. However, due to the unique structure of formal languages, they are represented as three-dimensional real number matrices, which cannot be processed using common network architectures. There is an urgent need for research into the embedding and encoding of formal languages.

We propose a Neural-symbolic architecture to address these issues. **The neural part** is a hypergraph neural network built on the attention mechanism, consisting of a hypergraph encoder and a theorem predictor. The former transforms hypergraph structural data into serialized hypernodes and adjacency matrices, then utilizes the convergence effect of the attention mechanism to embed variable-length serialized data into fixed-length real-number matrices and merge them to obtain the hypertree encoding. The latter uses a task-specific decoder that receives the hypertree encoding and the problem-solving target encoding to predict the theorems required for solving geometric problems. We first use a self-supervised task to pretrain the hypergraph encoder, forcing it to retain as much semantic information of formalized statements as possible during the encoding stage. It is then used as part of the theorem predictor for end-to-end training. **The symbolic part** is a symbolic formal system built on FormalGeo, which can construct the process of GPS as a directed hypertree with conditions as hypernodes and theorems as hyperedges. This symbolic system can validate and apply the theorems predicted by the neural part, perform geometric relational reasoning and algebraic equation solving, update the state of the solution hypertree, and ultimately realize traceable, verifiable, and interpretable automatic GPS.

Our contributions are summarized as follows:

1. We propose a neural-symbolic architecture for the automatic GPS. The neural part learns how to solve geometric problems from experience, predicting the general direction for problem-solving; the symbolic part performs strict geometric relational reasoning and algebraic equation solving to ensure the correctness of the solving process. We further propose the PAC cycle, clarifying the interaction between the neural part and the symbolic part.

2. We conducted experiments on the formalgeo7k dataset, ultimately achieving a step-wised accuracy of 87.65% and an overall accuracy of 85.53%. Additionally, we conduct ablation experiments on the training method and model architecture of HyperGNet.

2 Preliminaries

This section outlines the definition of the problem and models the problem-solving process. We focus on the formal representation and resolution of plane geometric problems.

2.1 Problem Definition

Traditional geometric problems encompass both image and text, serving as the foundation for problem representation. This necessitates the consideration of cross-modal alignment between the problem’s information components. Our focus lies in the resolution of geometric problems, employing formalized language for their description. The benefit of such formal representation lies in its ability

to construct structural information among geometric conditions, thus aiding in subsequent problem-solving.

We abstract geometric problems into a collection of known conditions and problem-solving targets. Furthermore, we conceptualize the problem-solving process in geometry as the application of a series of theorems. Consequently, the process of solving geometric problems can be constructed into a hypertree. The basic terms in our framework are defined below:

Definition 1 - Conditions (C): Conditions represent a collection of known prerequisites. These conditions encompass geometric and quantitative relationships, such as "RightTriangle(ABC)" and "Equal(LengthOfLine(AB),1)".

Definition 2 - Goal (G): The Goal symbolizes the objective of solving a geometric problem. The solving objective can be considered a special form of condition, such as "Value(MeasureOfAngle(ABC))".

Definition 3 - Theorems (T): Theorems constitute pre-defined prior knowledge. A theorem comprises a set of premise conditions and a set of conclusion conditions, both of which are collections of conditions. For instance, the parallel's transitivity can be expressed as "Parallel(AB,CD) & Parallel(CD,EF) \rightarrow Parallel(AB,EF)". The collection of all such theorem definitions forms the Prior Knowledge Base TKB .

Definition 4 - Solution Hypertree (H): The Solution Hypertree is a directed hypergraph with known conditions as hypernodes and applied theorems as hyperedges, describing the process of solving geometric problems. It is defined as $H = (C, T, G)$. A successful application of a theorem can add several new hypernodes to the hypertree and construct a new hyperedge from a set of premise conditions to a set of new conclusion conditions.

Based on these definitions, we can describe the task as: Given a formal representation of a geometric problem, construct it into a hypertree h . The initial hypertree h_0 contains only several discrete initial known condition nodes. Our task is to provide a sequence of theorem, where each application of t_i adds new hyperedges and hypernodes to h_{i-1} and extends h_{i-1} to h_i , ultimately constructing a reachable path from the initial set of known conditions to the problem-solving goal.

2.2 Predict-Apply Cycle

We have constructed a system comprising a formal environment and a neural agent to accomplish the aforementioned task. This system's dynamic process involves an interaction of two parts, which we refer to as the *Predict-Apply Cycle*, as illustrated in Fig. 1. The agent acquires the current solution hypertree h_{i-1} of the geometric problem and predicts the theorem t_i required for solving the problem. The formalized environment then receives and applies the theorem t_i , adding new hyperedges and hypernodes, thereby updating h_{i-1} to h_i . This interactive process is repeated continuously until the problem is solved or the hypertree ceases to update. The algorithm is described in Alg. 1.

Algorithm 1 Predict-Apply Cycle

Input: *problem*: geometric problems described using formalized language.
Output: *theorem_seqs*: theorem sequence for problem solving.
Initialize *env* and *agent*.
Initialize *theorem_seqs* as *None*.
Initialize *applied* as *True*.
env.init_hypertree(problem)
while *applied* **do**
 hypertree \leftarrow *env.get_hypertree()*
 theorem \leftarrow *agent.predict(hypertree)*
 applied \leftarrow *env.apply(theorem)*
 if *env.solved* is *True* **then**
 theorem_seqs \leftarrow *env.get_theorem_seqs()*
 break
 end if
end while

3 Neural-Symbolic Solver

This section introduces our proposed neural-symbolic architecture, which includes a symbolic formal system built on FormalGeo and a hypergraph neural network based on attention mechanisms.

3.1 Symbolic System

Existing work has failed to establish a consistent, traceable, and extensible formal system. To address this issue, we have developed a geometric symbolic formal system based on FormalGeo. FormalGeo employs geometric definition language to define the formal system and uses condition declaration language to declare the topological structure of geometric problems, known conditions, and problem-solving objectives. It transforms the application process of theorems into the execution process of geometric predicate logic enabling traceable relational reasoning and algebraic equation solving. This system bridges the gap between humans and computers, ensuring that the problem-solving process is both readable and interpretable.

The known conditions of geometric problems are stored as quintuples comprising condition ID, condition type, condition body, premises, and theorem. Based on the premises and theorems of geometric conditions, we group and structure these conditions, organizing them into a hypergraph with the condition body as hypernodes and theorem as hyperedges. Each hypernode has only one set of premises, connected by a theorem hyperedge, making it a hypertree. An example of solution hypertree is shown in Fig. 1 is illustrated in Fig. 2.

3.2 HyperGNet Architecture

Our task is to create a neural agent that can predict the theorems needed in the next step of solving a problem based on the hypertree h given by the current

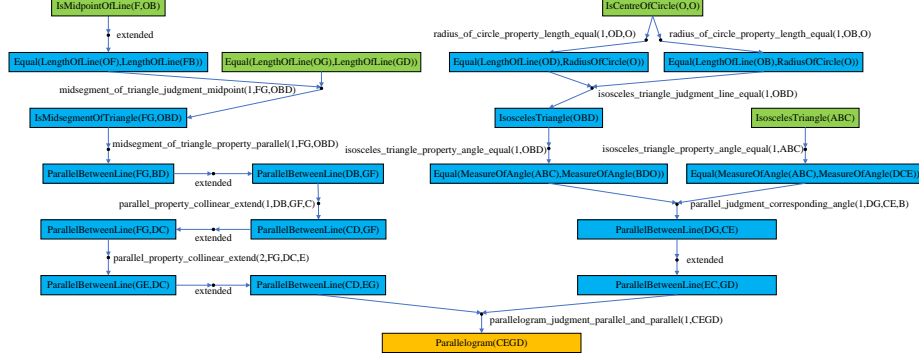


Fig. 2. An example of solution hypertree. The green nodes represent the known conditions, the orange nodes are the problem-solving objectives, and the blue nodes are added during the problem-solving process through the application of theorems. For layout considerations, hypertree in this image omits many details.

formal environment. This requires encoding the hypergraph into a real-number vector form that neural networks can understand. The Transformer [29] and its derivative network structures are considered powerful neural network architectures for modeling sequential data but are not capable of directly processing graph-structured data. Inspired by Graphormer [35], we first decompose the semantic and structural information in graph-structured data into a serialized form, and then input it into the network at appropriate positions for processing.

For a directed hypergraph containing n hypernodes, we can uniquely represent the hypergraph using the hypernode vector c and the hypergraph adjacency matrix $T_{n \times n}$. The elements c_i in c represent conditional declaration sentences, composed of predicates and individual words. To facilitate input into the neural network for processing, we need to embed c_i into an m -dimensional vector, ultimately allowing hypernodes to be represented by the matrix $N_{n \times m}$. The adjacency matrix $T_{n \times n}$ is an extremely sparse matrix, where the element t_{ij} indicates whether there is a hyperedge connecting hypernode a_i and hypernode a_j . Each row vector t_i of T contains the connectivity information between hypernode a_i and all other hypernodes. To obtain an overall representation of the hypergraph, combining hypernode information and the adjacency matrix, an intuitive method is to embed the i -th row of the adjacency matrix into an m -dimensional vector as another representation of the hypernode a_i , and add it to the i -th row of C . When embedding the row vectors of the adjacency matrix T into m -dimensional vectors, due to its extremely sparse nature, traditional methods [22, 16] would result in a large number of neurons being idle and wasted. For each row vector t_i of T , we first record the indices of each non-zero element to form vector s_i ; then, we remove the zero elements from t_i to form a new row vector e_i . Embedding s and e into m -dimensional vectors yields the structural information representa-

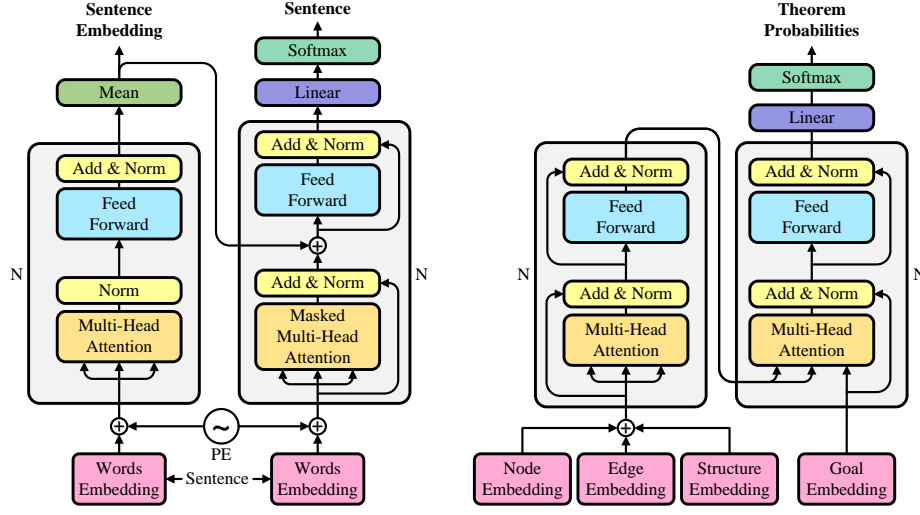


Fig. 3. The overall architecture of HyperGNet. The left side presents a hypertree encoder that utilizes the convergent effects of the attention mechanism, forming the input encoding part of the network on the right. The right side features a theorem predictor based on task-specific attention, known as HyperGNet.

tion $S_{n \times m}$ and the semantic information representation $E_{n \times m}$ for hyperedges. The input to the theorem prediction network can be obtained through Eq. 1:

$$H_{n \times m} = N_{n \times m} + E_{n \times m} + S_{n \times m} \quad (1)$$

As shown in Fig. 3, HyperGNet adopts an encoder-decoder architecture. We use the standard transformer encoder modules to construct the HyperGNet encoder, and task-specific attention to construct HyperGNet Decoder. As previously mentioned, the solving process of geometric problems can be described with a hypertree $H = (C, T, G)$, where C and T are processed through N layers of encoder modules to obtain the hypertree encoding $H_{n \times m}^{(N)}$. For the GPS target G , it can be considered as a special hypernode and embedded into an m -dimensional vector g . After linear transformation of $H_{n \times m}^{(N)}$ and g , a task-specific attention layer and a feed-forward layer with RELU are used to extract key information relevant to problem-solving, as shown in Eq. 2, Eq. 3, Eq. 4, Eq. 5 and Eq. 6.

$$Q = gW^{(Q)} \quad (2)$$

$$K = H_{n \times m}^{(N)} W^{(K)} \quad (3)$$

$$V = H_{n \times m}^{(N)} W^{(V)} \quad (4)$$

$$\text{TSAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (5)$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (6)$$

For the solution hypertree H , the application process of theorems can be constructed as a directed acyclic graph. In the intermediate stages of GPS, there are multiple alternative theorems. Therefore, we model the theorem prediction task as a multi-class classification task, calculating the cross-entropy loss for each predicted theorem value. The loss function is shown in Eq. 7, where x is the predicted value, y is the true value, σ is the sigmoid activation function, and N is the number of defined theorems.

$$\text{Loss}(x, y) = -\frac{1}{N} \sum_{i=1}^N [y_i \cdot \log(\sigma(x_i)) + (1 - y_i) \cdot \log(1 - \sigma(x_i))] \quad (7)$$

3.3 Hypertree Encoder

In the previous section, we mentioned embedding c_i , s_i , e_i , and G as m -dimensional vectors, which is a type of variable-length sentence embedding problem. In these problems, we need to embed sentences of length n ($n > 1$) into a fixed-dimensional vector representation. Traditional methods [16] for encoding variable-length sentences typically involve padding or truncating the sentences to a uniform length before feeding them into a neural network for processing. Sentence embedding methods based on the attention mechanism take the average of the row vectors of the sentence's encoding matrix [33] or use the first word as the sentence's embedding representation after obtaining the encoding matrix [22]. These sentence embedding methods will more or less lose the original semantic information of the sentences.

The attention mechanism is essentially a special method of weighted averaging that causes individuals to converge towards the overall mean. In the transformer architecture, residual connections [14] maintain the differences in the representations of different words within a sentence. In the field of graph neural network research, this phenomenon is known as oversmoothing [4], where each node in the graph becomes increasingly similar as it receives global information. This is typically viewed as a phenomenon to be avoided, but in this paper, we utilize this phenomenon to achieve the embedding of variable-length sentences.

As shown in Fig. 3, we adopt the standard transformer architecture to implement the embedding of variable-length sentences. We removed the residual connections from encoder, which results in each word in the sentence converging to the same representation after multiple attention layers, and this representation is taken as the overall representation of the sentence. We set up a self-supervised training task using cross-entropy loss, as shown in Eq. 8.

$$L(x, y) = -\log \left(\frac{\sum_{i=1}^C y_i \exp(x_i)}{\sum_{i=1}^C \exp(x_i)} \right) \quad (8)$$

4 Experiments

This section presents the performance of our neural-symbolic architecture on formalgeo7k dataset. We compare and analyze the differences in solving time and step between search-based methods and the approach proposed in this paper. Additionally, we conduct ablation experiments on the training method and model architecture of HyperGNet.

4.1 Dataset

The latest advancements in AI are driven by the combination of data and computational power, with data playing a crucial role in the performance of AI models. We have constructed the formalgeo7k dataset and conducted a comparative analysis with other existing datasets, as shown in Tab. 2.

Table 2. An overview of geometry-related datasets.

Datasets	Tasks Formalized		Size	Comparative Metrics					
				GT	GI	NT	PT	ET	OS
GEOS [25]	GPS	FL	186	✓	✓	✓			✓
GEOS++ [23]	GPS	FL	1,406	✓	✓	✓			✓
GeoShader [1]	GPS	FL	102	✓	✓	✓			✓
Geometry3K [18]	GPS	FL	3,002	✓	✓	✓			✓
GeoQA [6]	GPS	Program	5,010	✓	✓	✓			✓
GeometryQA [28]	GPS	Program	1,398	✓		✓			✓
GeoRE [36]	GRE	FL	12,901	✓		✓			✓
GeoQA+ [3]	GPS	Program	7,528 (5010 from [6])	✓	✓	✓			✓
UniGeo [5]	GPS	Program	14,541 (4998 from [6])	✓	✓	✓	✓		✓
PGDP5K [13]	GDP	FL	5,000 (1813 from [18])		✓	✓			✓
PGPS9K [39]	GPS	Program	9,022 (2891 from [18])	✓	✓	✓			✓
Tri300 [31]	GPS	FL	300 (DA to 1,000)	✓		✓			✓
Ours [40]	GPS	FL	6981 (DA to 133,818)	✓	✓	✓	✓	✓	✓

* GPS denotes geometric problem solving. GRE denotes geometric relation extraction. GDP denotes geometric diagram parsing. FL denotes formal language. DA denotes data augmentation. The 5 comparative metrics are geometric text, geometric diagram, numerical target, proving target, extensible, and open source.

We conducted experiments on the unexpanded formalgeo7k dataset, partitioning it into a training set, validation set, and test set at a ratio of 3:1:1. The problem-solving process for each question involves the application of multiple

theorems and can be constructed as a directed acyclic graph (DAG) of theorems. Any theorem sequence obtained by traversing this DAG can solve the problem. We randomly traversed the DAG and obtained each step’s problem state and the set of applicable theorems, yielding training data pairs (hypertree, theorems). This process ultimately generated 20,571 training data entries (from 4,079 problems), 7,072 validation data entries (from 1,370 problems), and 7,046 test data entries (from 1,372 problems).

4.2 Training Method

We initially utilize a self-supervised method to pre-train the hypertree encoder, ensuring it achieves the highest accuracy on the validation set. Subsequently, the hypertree encoder is incorporated as part of HyperGNet, facilitating end-to-end training.

In the model hyperparameters as presented in Figure 3, we set the hidden dimension d_{model} to 256, the layer count N to 4, and the number of attention heads h to 4. The total parameter size of HyperGNet amounts to 338.55 MB. During the model’s training phase, we optimize the model parameters using the Adam algorithm, with a batch size of 64, training for 50 epochs, and a learning rate of 10^{-5} . Executing a single training session (both hypertree encoder pretraining and HyperGNet training) on an GeForce RTX 4090 approximately requires 30 minutes.

4.3 Evaluation

We compared the Greedy Forward Search method (FS) with the heuristic search method that integrates HyperGNet in terms of problem-solving success rate, as shown in Tab. 3.

We ran the search-based solving algorithm using different strategies. BFS stands for Breadth-First Search. DFS stands for Depth-First Search. RS stands for Random Search, where a theorem is randomly selected from the candidate set. RBS stands for Random Beam Search, where k theorems are selected from the candidate set, with k being the size of the beam. These are the search strategies for FS. For the search strategies of HyperGNet, NB stands for Normal Beam, where the probability of theorem selection is calculated based on the predicted probability of each theorem given by the network, as well as the probability of each beam head, as shown in Eq. 9 and Eq. 10. The k highest probability (beam, theorem) pairs are selected. GB stands for Greedy Beam, which, based on NB, add applicable theorems to beam, removing any theorems that cannot be applied, until the size of the beam reaches k .

$$p^{(\text{net},i)} = \text{HyperGNet}(h_i) \quad (9)$$

$$P = \{p_{i,j} | p_{i,j} = p_j^{(\text{net},i)} \cdot p_i^{(\text{beam})}\} \quad (10)$$

As shown in Tab. 3, under a timeout setting of 30s, the heuristic search combined with HyperGNet achieved nearly a 5-fold increase in problem-solving success rate compared to greedy forward search. Comparing the two heuristic search strategies, NB and GB, it can be observed that the NB strategy is more efficient. However, because it does not check whether the theorem is successfully applied, it leads to a progressively smaller beam size, resulting in a lower problem-solving success rate.

Table 3. An overview of the problem-solving success rates.

Method	Strategy	Beam Size	Timeout	Acc (%)		
				Solved	Unsolved	Timeout
FS	BFS	/	30	11.46	3.87	84.67
FS	DFS	/	30	15.40	5.44	79.15
FS	RS	/	30	17.91	5.95	76.15
FS	RBS	20	30	13.90	11.82	74.28
HyperGNet	NB	5	30	62.18	28.15	9.67
HyperGNet	GB	5	30	75.00	0.50	24.50
HyperGNet	GB	5	600	85.53	2.22	12.25

* The data for the FW method is sourced from [40]. The original data had a timeout of 600, and we recalculated the solution times for the FW method in this table based on a timeout of 30.

In accordance with the length of the theorems required for problem-solving, we roughly categorize the difficulty of the questions into 6 levels, denoted as l_1 ($length \leq 2$), l_2 ($3 \leq length \leq 4$), l_3 ($5 \leq length \leq 6$), l_4 ($7 \leq length \leq 8$), l_5 ($9 \leq length \leq 10$), l_6 ($length \geq 11$). The problem-solving success rates for geometric problems of varying difficulties are as shown in Tab. 4.

Table 4. Details of the problem-solving success rates.

Method	Strategy	Beam Size	Timeout	Acc (%)						
				Total	L_1	L_2	L_3	L_4	L_5	L_6
FS	BFS	/	30	11.46	26.56	8.38	0.39	0.00	0.00	0.00
FS	DFS	/	30	15.40	31.54	15.41	1.57	0.60	0.00	1.69
FS	RS	/	30	17.91	37.97	14.59	2.76	2.99	0.00	1.69
FS	RBS	20	30	13.90	32.57	8.65	1.18	1.20	0.00	0.00
HyperGNet	NB	5	30	62.18	82.57	65.14	51.57	46.71	20.31	11.86
HyperGNet	GB	5	30	75.00	91.49	80.27	68.11	61.08	29.69	25.42
HyperGNet	GB	5	600	85.53	95.44	89.46	84.25	77.84	50.00	45.76

* The data for the FW method is sourced from [40].

Observing the data in Tab. 4, it's evident that the heuristic search combined with HyperGNet significantly outperforms greedy search in terms of problem-solving success rate across all levels of difficulty in geometric problems. Par-

ticularly, the disparity becomes more pronounced as the difficulty level of the problems increases. Heuristic search avoids the examination of theorems unrelated to problem-solving, which substantially reduces the time required to find a solution, as illustrated in Fig. 4.

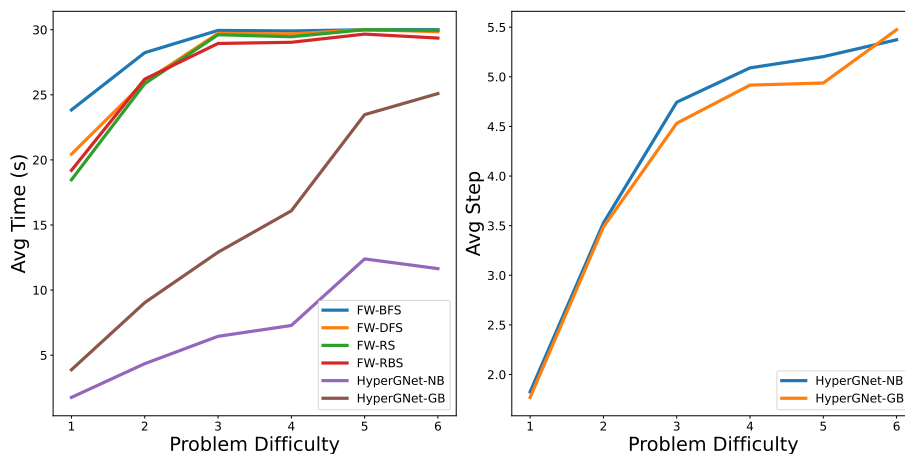


Fig. 4. Average solving time and solving step.

We compared the differences in search steps between HyperGNet’s NB and GB strategies, as depicted in Fig. 4. When the problems are of lower difficulty, the search steps of both strategies are roughly similar. However, as the difficulty increases, the search steps of the NB strategy are generally slightly higher than those of the GB strategy. This is because the NB strategy is more likely to discard the correct theorem selection at each step. As the difficulty of the problem escalates and the theorem sequence becomes longer, the accumulation of errors can lead to the inability to solve the problem successfully, resulting in termination only when it timeout.

4.4 Ablation Study

We conduct ablation experiments on the training method and model architecture of HyperGNet, as shown in Tab. 5. The term *-w/o Pretrain* indicates the removal of the pre-training step, proceeding directly to end-to-end training. *-w/o Hypertree* denotes not using hypergraph structural data; This is implemented by, without altering the network architecture, removing the hyperedge information and inputting only the node information as sequential data into the network.

We can observe that as the beam size increases, the step-wised theorem prediction accuracy and problem-solving success rate also improve. However, this

also results in increased time and steps required. To test the effectiveness of pre-training, we removed the pretraining stage and found that the problem-solving success rate fell by about 5 percentage points under a beam size of 5. Therefore, using pre-training to force the encoder to retain the semantic information of formalized statements is necessary. To test the importance of graph-structured data, we eliminated the hyperedge information. The experimental results show that while the stepwise theorem prediction success rate increased after the removal of graph structure data, the final problem-solving success rate decreased. This suggests that the historical information of theorem applications contributes to improving the final problem-solving success rate.

Table 5. Ablation study.

Method	Beam Size	Step-Wised Acc (%)	Overall Acc (%)	Avg Time (s)	Avg Step
HyperGNet	1	63.03	30.23	0.96	2.62
	3	82.05	53.30	3.05	3.31
	5	87.65	62.18	5.47	3.46
-w/o Pretrain	1	62.80	27.15	0.86	2.57
	3	82.86	48.21	2.70	3.33
	5	88.66	57.95	4.86	3.50
-w/o Hypertree	1	62.48	29.66	0.80	2.55
	3	83.08	51.29	2.80	3.28
	5	89.24	60.67	4.73	3.38

* The data in the table were obtained using the NB strategy. *Step-Wised Acc* refers to the success rate of step-wised theorem prediction, while *Overall Acc* denotes the success rate of solving all the problems.

5 conclusions

This paper proposes a neural-symbolic architecture for solving formalized plane geometry problems. The neural part consists of a hypertree encoder and a theorem predictor based on the attention mechanism; the former encodes hypergraph structural data into real-number matrices, and the latter receives the hypertree encoding and predicts the theorems required for problem-solving. The symbolic part is a symbolic formal system built on FormalGeo, which can construct the process of GPS as a hypertree with conditions as hypernodes and theorems as hyperedges, realizing a traceable and verifiable automatic GPS. Finally, we conducted comparative and ablation experiments to verify the effectiveness of the architecture.

In the future, we plan to integrate reinforcement learning into this system to achieve GPS without human supervision and further enhance theorem prediction capabilities. In addition, we plan to attempt solving higher difficulty geometry problems, such as those of the International Mathematical Olympiad level.

Acknowledgement

This research was supported by NSFC Grant.12071282.

Contributions

Xiaokai Zhang: Conceptualization, Methodology, Coding, Writing – original draft & review & editing. **Na Zhu, Yiming He, Jia Zou, Cheng Qin, Yang Li:** Conceptualization, Methodology, Writing – review & editing. **Zhenbing Zeng:** Conceptualization, Methodology. **Tuo Leng:** Supervision, Funding acquisition, Conceptualization, Methodology, Writing – review & editing.

References

1. Alvin, C., Gulwani, S., Majumdar, R., Mukhopadhyay, S.: Synthesis of solutions for shaded area geometry problems. In: The Thirtieth International Flairs Conference (2017)
2. Buchberger, B.: Applications of gröbner bases in non-linear computational geometry. *Mathematical aspects of scientific software* pp. 59–87 (1988)
3. Cao, J., Xiao, J.: An augmented benchmark dataset for geometric question answering through dual parallel text encoding. In: Proceedings of the 29th International Conference on Computational Linguistics. pp. 1511–1520 (2022)
4. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 3438–3445 (2020)
5. Chen, J., Li, T., Qin, J., Lu, P., Lin, L., Chen, C., Liang, X.: Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 3313–3323. Association for Computational Linguistics (2022)
6. Chen, J., Tang, J., Qin, J., Liang, X., Liu, L., Xing, E., Lin, L.: Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 513–523 (2021)
7. Chou, S.C., Gao, X.S., Zhang, J.Z.: Automated production of traditional proofs in solid geometry. *Journal of Automated Reasoning* **14**(2), 257–291 (1995)
8. Chou, S.C., Gao, X.S., Zhang, J.Z.: A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning* **25**(3), 219–246 (2000)
9. Daniel, S., Leonardo, d.M., Kevin, B., Reid, B., Percy, L., Sarah, L., Freek, W.: Imo grand challenge (2019), <https://imo-grand-challenge.github.io/>
10. Gan, W., Yu, X., Zhang, T., Wang, M.: Automatically proving plane geometry theorems stated by text and diagram. *International Journal of Pattern Recognition and Artificial Intelligence* **33**(07), 1940003 (2019)
11. Gelernter, H.L.: Realization of a geometry theorem proving machine. In: IFIP congress. pp. 273–281 (1959)

12. Guo, F., Jian, P.: A graph convolutional network feature learning framework for interpretable geometry problem solving. In: 2022 International Conference on Intelligent Education and Intelligent Research (IEIR). pp. 59–64. IEEE (2022)
13. Hao, Y., Zhang, M., Yin, F., Huang, L.L.: Pgdp5k: A diagram parsing dataset for plane geometry problems. In: 2022 26th International Conference on Pattern Recognition (ICPR). pp. 1763–1769. IEEE (2022)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Jian, P., Guo, F., Wang, Y., Li, Y.: Solving geometry problems via feature learning and contrastive learning of multimodal data. *CMES-COMPUTER MODELING IN ENGINEERING & SCIENCES* **136**(2), 1707–1728 (2023)
16. Li, B., Zhou, H., He, J., Wang, M., Yang, Y., Li, L.: On the sentence embeddings from pre-trained language models. arXiv preprint arXiv:2011.05864 (2020)
17. Littman, M.L., Ajunwa, I., Berger, G., Boutilier, C., Currie, M., Doshi-Velez, F., Hadfield, G., Horowitz, M.C., Isbell, C., Kitano, H., et al.: Gathering strength, gathering storms: The one hundred year study on artificial intelligence (ai100) 2021 study panel report. arXiv preprint arXiv:2210.15767 (2022)
18. Lu, P., Gong, R., Jiang, S., Qiu, L., Huang, S., Liang, X., Zhu, S.c.: 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 (Volume 1: Long Papers). pp. 6774–6786 (2021)
19. Nevins, A.J.: Plane geometry theorem proving using forward chaining. *Artificial Intelligence* **6**(1), 1–23 (1975)
20. Ning, M., Wang, Q.F., Huang, K., Huang, X.: A symbolic characters aware model for solving geometry problems. In: Proceedings of the 31st ACM International Conference on Multimedia. p. 7767–7775. Association for Computing Machinery (2023)
21. Peng, S., Fu, D., Liang, Y., Gao, L., Tang, Z.: Geodrl: A self-learning framework for geometry problem solving using reinforcement learning in deductive reasoning. In: Findings of the Association for Computational Linguistics: ACL 2023. pp. 13468–13480 (2023)
22. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019)
23. Sachan, M., Dubey, K., Xing, E.: 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. pp. 773–784 (2017)
24. Sachan, M., Xing, E.: Learning to solve geometry problems from natural language demonstrations in textbooks. In: Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (SEM 2017). pp. 251–261 (2017)
25. Seo, M., Hajishirzi, H., Farhadi, A., Etzioni, O., Malcolm, C.: Solving geometry problems: Combining text and diagram interpretation. In: Proceedings of the 2015 conference on empirical methods in natural language processing. pp. 1466–1476 (2015)
26. Song, B., Xiong, G., Shen, Z., Zhu, F., Lv, Y., Ye, P.: Geometry problem solving based on counter-factual evolutionary reasoning. In: 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE). pp. 1–6. IEEE (2023)
27. Trinh, T.H., Wu, Y., Le, Q.V., He, H., Luong, T.: Solving olympiad geometry without human demonstrations. *Nature* **625**(7995), 476–482 (2024)

28. Tsai, S.h., Liang, C.C., Wang, H.M., Su, K.Y.: 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 (Volume 2: Short Papers). pp. 964–972 (2021)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* **30** (2017)
30. Wu, W.T.: On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica* **21**, 157–179 (1978)
31. Xiao, Z., Zhang, D.: A deep reinforcement learning agent for geometry online tutoring. *Knowledge and Information Systems* **65**(4), 1611–1625 (2023)
32. XTXMarkets: Artificial intelligence mathematical olympiad prize (aimo prize) (2023), <https://aimoprize.com/>
33. Yan, Y., Li, R., Wang, S., Zhang, F., Wu, W., Xu, W.C.: A contrastive framework for self-supervised sentence representation transfer. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. vol. 1
34. Yang, L., Gao, X.S., Chou, S.C., Zhang, J.Z.: Automated production of readable proofs for theorems in non-euclidean geometries. In: *Automated Deduction in Geometry: International Workshop on Automated Deduction in Geometry Toulouse, France, September 27–29, 1996 Selected Papers 1*. pp. 171–188. Springer (1997)
35. Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.Y.: Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems* **34**, 28877–28888 (2021)
36. Yu, W., Wang, M., Wang, X., Zhou, X., Zha, Y., Zhang, Y., Miao, S., Liu, J.: 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)
37. Zhang, J.Z., Chou, S.C., Gao, X.S.: Automated production of traditional proofs for theorems in euclidean geometry i. the hilbert intersection point theorems. *Annals of Mathematics and Artificial Intelligence* **13**(1-2), 109–137 (1995)
38. Zhang, M.L., Li, Z.Z., Yin, F., Liu, C.L.: Lans: A layout-aware neural solver for plane geometry problem. *arXiv preprint arXiv:2311.16476* (2023)
39. Zhang, M.L., Yin, F., Liu, C.L.: 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-23*. pp. 3374–3382. International Joint Conferences on Artificial Intelligence Organization (2023)
40. Zhang, X., Zhu, N., He, Y., Zou, J., Huang, Q., Jin, X., Guo, Y., Mao, C., Zhu, Z., Yue, D., Zhu, F., Li, Y., Wang, Y., Huang, Y., Wang, R., Qin, C., Zeng, Z., Xie, S., Luo, X., Leng, T.: Formalgeo: The first step toward human-like imo-level geometric automated reasoning (2023)