Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.023243

ARTICLE

Solving Geometry Problems via Feature Learning and Contrastive Learning of Multimodal Data

Pengpeng Jian[1], Fucheng Guo[1,*], Yanli Wang[2] and Yang Li[1]

[1]North China University of Water Resources and Electric Power, Zhengzhou, 450046, China

[2]Henan University of Economics and Law, Zhengzhou, 450046, China

*Corresponding Author: Fucheng Guo. Email: guofucheng@stu.ncwu.edu.cn

Received: 15 April 2022 Accepted: 23 September 2022

ABSTRACT

This paper presents an end-to-end deep learning method to solve geometry problems via feature learning and contrastive learning of multimodal data. A key challenge in solving geometry problems using deep learning is to automatically adapt to the task of understanding single-modal and multimodal problems. Existing methods either focus on single-modal or multimodal problems, and they cannot fit each other. A general geometry problem solver should obviously be able to process various modal problems at the same time. In this paper, a shared feature-learning model of multimodal data is adopted to learn the unified feature representation of text and image, which can solve the heterogeneity issue between multimodal geometry problems. A contrastive learning model of multimodal data enhances the semantic relevance between multimodal features and maps them into a unified semantic space, which can effectively adapt to both single-modal and multimodal downstream tasks. Based on the feature extraction and fusion of multimodal data, a proposed geometry problem solver uses relation extraction, theorem reasoning, and problem solving to present solutions in a readable way. Experimental results show the effectiveness of the method.

KEYWORDS

Geometry problems; multimodal feature learning; multimodal contrastive learning; automatic solver

1 Introduction

Automatically solving geometry problems is a long-standing challenge in artificial intelligence that continues to attract research interest [1–3], which has focused such as on angle and length calculation. Geometry problem solving has been modeled as a process of relation extraction and reasoning [4–6], using a syntax-semantic model to extract geometric relations from problem text, and sending them to an expert system for reasoning and solution. It can also be modeled in terms of submodular optimization [7,8]. The wide application of deep learning methods in natural language processing, computer vision, and vision-language interaction makes it possible to solve geometry problems based on deep learning [9–12]. Whether focusing on problem parsing and relation reasoning or on interpretable presentation, past work has provided important insights.

However, solving geometry problems requires the mapping of human-readable text and visual images into machine-understandable logical forms, followed by reasoning and solution, and not simply by pattern recognition and matching or end-to-end classification [12]. Geometry problems are often presented in multimodal forms such as text, images, and image-text pairs, which brings challenges to their unified representation, which contains multimodal data such as text, symbols, formulas, and images. The different statistical characteristics of modal data lead to a lack of semantic correlation between them. Different modal representations of an entity contain both shared and unique information. If this information is not semantically related, it will affect the understanding of multimodal data, hindering understanding and automatic solution. There is a semantic gap in the multimodal understanding process of geometry problems. Information can describe the same object or event from different angles, and it suffers from a lack of correlation. Although the information on multimodal processing and fusion is rich and detailed [13–15], it is difficult to apply to the solution of geometry problems.

This paper presents an end-to-end, fully automated deep learning method to solve geometry problems via feature learning and contrastive learning of multimodal data. Solving geometry problems by deep learning has the steps of feature extraction, fusion, and reasoning. A shared feature-learning model of multimodal data learns the unified feature representation of text and images, which can solve the heterogeneity problem of multimodal geometry problems. A contrastive learning model of multimodal data enhances the semantic relevance between multimodal features and maps them into a unified semantic space, which can effectively adapt to both single-modal and multimodal downstream tasks. Based on the feature extraction and fusion of multimodal data, a geometry problem solver is proposed, adopting a shared encoder-decoder structure to generate solution sequences. A shared encoder realizes the deep understanding of text and/or images by a multi-headed self-attention mechanism. For multimodal problems, a multilayer Transformer realizes the interaction between cross-modal features. For single-modal problems, the Transformer can adaptively attend to single-modal data. Multimodal contrastive learning addresses the lack of semantic correlation between multimodal data, which can attract relevant text and/or image features, repel irrelevant features in the representation space, and realize the semantic alignment of multimodal data. A shared decoder decodes single-modal contrastive vectors or a series sequence of multimodal contrastive vectors according to the input of the encoder. The encoder and decoder can be cascaded to obtain deeper implicit information. The representation of the decoder is transferred to task-specific heads for geometry relations extraction, theorem reasoning, and problem solution. The proposed algorithm can produce readable solutions. Target programs can assist in problem solution and model diagnosis. Similar to machine translation, beam search produces a better target program. Multiple auxiliary tasks, including puzzle recovery and image element identification, improve the performance of the image embedder. Experiments on two public geometry problem datasets demonstrate the effectiveness of the proposed algorithm.

The main contributions of this paper are as follows:

- An end-to-end deep learning method to solve geometry problems is proposed, whose input is a geometry problem with text and/or images, and which produces a readable solution procedure. This single framework adaptively solves single-modal and multimodal geometry problems without modifying the model structure;
- We propose a shared feature-learning model of multimodal data for pretraining geometry problem text and/or images. Because the architecture of multimodal feature learning utilizes

a similar network of self-attention masks, extracted multimodal features can be formatted in a unified feature representation, so as to address heterogeneity in multimodal geometry problems;

- A contrastive learning model of multimodal data is proposed to enhance the semantic relevance between multimodal features and map them into a unified semantic space, which can effectively adapt to both single-modal and multimodal downstream tasks.

The remainder of this paper is organized as follows. Section 2 discusses related research. Section 3 presents our proposed method. Experimental results are given in Section 4, and conclusions are drawn in Section 5.

2 Related Work

Research on feature and contrastive learning can be categorized as single-modal, multimodal, or cross-modal. Single-modal methods are only used to train single-modal tasks, such as for text or image data. Multimodal methods relate to training multimodal tasks, such as for image-text pair data, and cross-modal methods concentrate on the compatibility between single-modal and multimodal tasks.

Single-Modal Learning. Single-modal learning methods focus on the extraction and analysis of text or image features. Most natural language text learning methods are based on the multilayer Transformer architecture. For instance, BERT [16] uses bidirectional representations from Transformers with 24 layers, UniLM [17] uses three types of language modeling tasks to train a shared multilayer Transformer network, RoBERTa [18] optimizes BERT for key hyperparameters and training data size, XLNet [19] uses generalized autoregressive pretraining for a multilayer Transformer, and BART [20] is a language-generating model with a denoising autoencoder. Visual image learning methods are mainly based on the multilayer CNN architecture, such as VGG [21], with 19 weight layers of convolutional networks with very small convolution filters; and ResNet [22], with a 152-layer residual learning framework with less complexity and better performance than VGG. For single-modal problems in different disciplines, both text and image data have the problems of high noise and redundancy. Traditional single-modal methods can perform well only in text or image training; they lack the ability to process multimodal tasks with both text and images, and extracted local and global features cannot be effectively adapted to downstream tasks.

Multimodal Learning. Multimodal learning methods are used, especially in the training of image-text data pairs. There are both one-channel and multi-channel methods. The former uses a single Transformer architecture to train a series sequence of multimodal data. UNITER [23] uses a multilayer Transformer to learn a cross-modal contextualized embedding between visual regions and textual tokens, VisualBERT [24] aligns tokens of the input text and regions in the input image implicitly by Transformer, and VL-BERT [25] extends the Transformer to take both visual and linguistic embedded features as input. Multi-channel methods use multiple Transformer architectures to train different single-modal data. For instance, ViLBERT [26] extends the BERT architecture to parallel models for text and image processing, and uses co-attentional Transformer layers to interact with separate streams. MAGIC [27] leverages three parallel decoders to generate responses in different media. MAS [28] is proposed for awakening the robot without wake words by audiovisual consistency detection and semantic talking intention inference. LARCH [29] performs a conversational image search through a multimodal hierarchical graph-based neural network, multi-form knowledge embedding memory network, and gated neural network. These methods provide some useful ideas for solving geometry problems. On the observation that multimodal information fusion in one-channel methods is proposed earlier than multi-channel methods, which can achieve better performance. Limited to the use of image-text data pairs for training, these multimodal learning methods can only train small-scale data, and cannot be effectively adapted to single-modal downstream tasks.

Cross-Modal Learning. Cross-modal learning methods use text, image, and image-text pairs simultaneously as training data to learn the unified semantic representation of text and images. AttnGAN [30] uses a cross-modal attention learning model, which allows attention-driven, multi-stage refinement for fine-grained text, and can synthesize fine-grained details in different subregions by focusing on relevant words in the natural language description. Contrastive learning is effective for self-supervised representation learning, which attracts relevant text and/or image features to each other in the representation space, and repels irrelevant features, so as to realize the semantic alignment of cross-modal data [31]. UNIMO [32] can effectively adapt to both single-modal and multimodal understanding and generation tasks by using multilayer self-attention Transformers to learn unified semantic representations of cross-modal information, so they can be effectively adapted to downstream tasks.

Geometry Problem Solving. Geometry problem solving efforts are increasing. An automated system, GEOS, solves geometry problems by combining text understanding and image interpretation, where element relations in the text are obtained by machine learning, and the primitive and its relations in the image are obtained by maximizing the consistency of image and text data. An interpretable geometry problem solver (Inter-GPS) parses the problem text and image into formal language by rule-based text parsing and neural object detection, and incorporates theorem knowledge as conditional rules and performs symbolic reasoning [9]. A neural geometric solver (NGS) addresses geometry problems by comprehensively parsing multimodal information and generating interpretable programs [10]. The above work only focuses on multimodal problem solving, and feature representation needs to improve.

This paper proposes an end-to-end deep learning method for solving geometry problems via the feature learning and contrastive learning of Multimodal data, where both the single-modal geometry problems, such as text or images and Multimodal geometry problems, such as image-text pairs are considered. The proposed method shows good performance in solving geometry problems with both single-modal and Multimodal tasks.

3 Proposed Method

We present the proposed method to solve geometry problems via feature learning and contrastive learning of multimodal data. As shown in Fig. 1, the overall framework of this paper is to integrate the encoder and decoder network structure of a multilayer converter. The converter improves the feature representation and problem-solving ability of multimodal data, which can help build an end-to-end deep learning method for solving geometry problems. Figs. 2 and 3 show the two modules for feature representation. This paper considers the text and/or image of a geometry problem as input. The multilayer Transformer in the encoder converts the text and/or image to implicit context features, and features at different levels are semantically aligned and mapped into a unified semantic space through feature and contrastive learning of multimodal data, so as to form a unified feature representation. The multilayer Transformer decodes the unified multimodal features according to task-specific learning embedding to form the shared features of multi-tasks, which are transferred to the head of specific learning tasks, which are learned to solve problems. The multimodal and single-modal learning networks respectively solve image-text and natural language text geometry problems; it should be noted that the networks share the same structure.

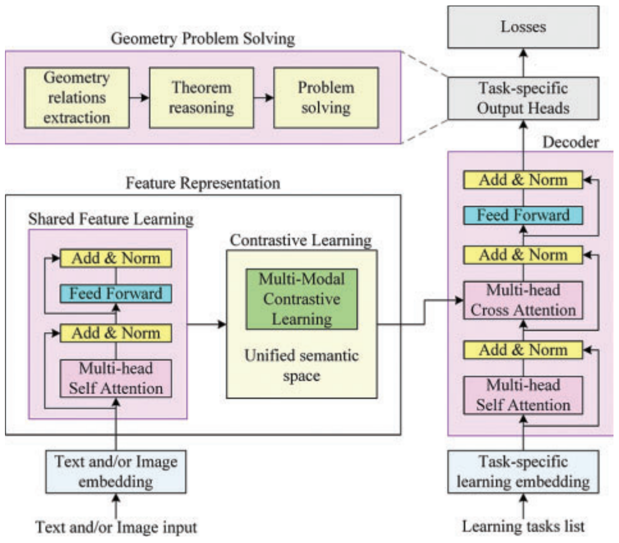


Figure 1: Overall architecture of the proposed method

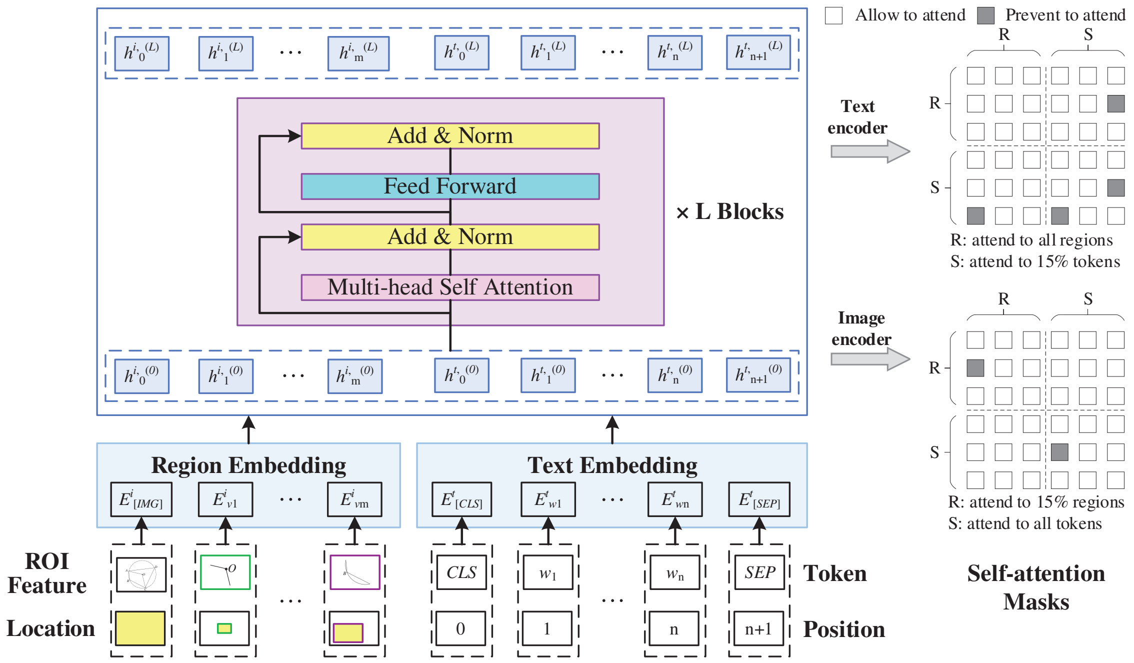


Figure 2: Shared feature-learning model of multimodal data

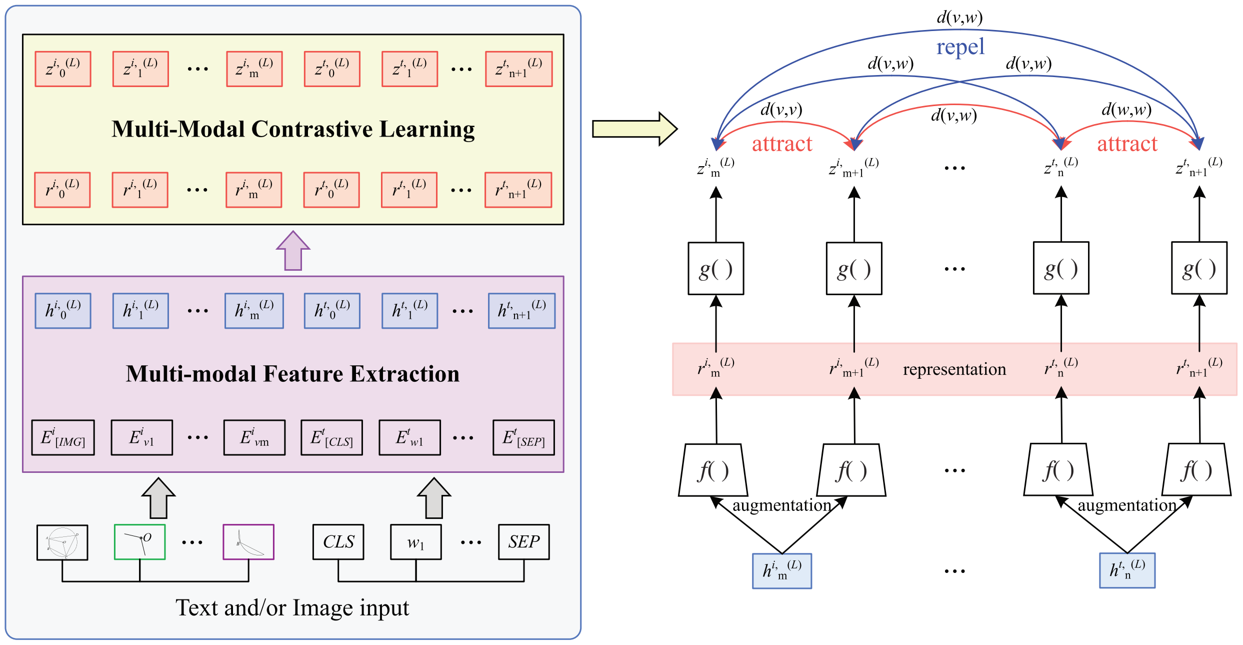


Figure 3: Contrastive learning model of multimodal data which are text and/or image

3.1 Multimodal Feature Learning

Due to the heterogeneity between multimodal geometry problems, the feature distribution of data in each mode is different, which brings difficulties to feature extraction. To model the feature extraction of multimodal problems and deal with various modal data, we establish a shared feature-learning model, which can extract features in single-modal text or image data as well as multimodal image-text data. The model masks text and image features, and realizes the interaction between multimodal data through multilayer Transformers, so as to produce better feature expression of multimodal data and support downstream multimodal tasks.

Fig. 2 shows the shared feature-learning model of multimodal data. For problem text, [CLS] represents the classification mark at the beginning of a sequence, and [SEP] represents the separator mark at the end. For a geometry image, [IMG] represents the classification mark of the start sequence. For the text and image input of multimodal geometry problems, we obtain the respective text tag feature and image region feature embedding sequences, $W = \{E^t_{[CLS]}, E^t_{w_1}, \cdots, E^t_{w_n}, E^t_{[SEP]}\}$ and $V = \{E^i_{[IMG]}, E^i_{v_1}, \cdots, E^i_{v_m}\}$, using the feature embedding module of multimodal data. Then, if the feature learning task is a multimodal problem such as image-text, the two feature embedding sequences are connected into a feature embedding sequence of pairs $VW = \{E^i_{[IMG]}, E^i_{v_1}, \cdots, E^i_{v_m}, E^t_{[CLS]}, E^t_{w_1}, \cdots, E^t_{w_n}, E^t_{[SEP]}\}$, which is input to the multilayer self-attention converter to learn the multimodal feature representation of the text mark and image region, outputting the text feature representation $h^t = \{h^{t,(L)}_0, h^{t,(L)}_1, \cdots, h^{t,(L)}_n, h^{t,(L)}_{n+1}\}$, image feature representation $h^i = \{h^{i,(L)}_0, h^{i,(L)}_1, \cdots, h^{i,(L)}_m\}$, or image-text feature representation $h = \{h^{i,(L)}_0, h^{i,(L)}_1, \cdots, h^{i,(L)}_m, h^{t,(L)}_0, h^{t,(L)}_1, \cdots, h^{t,(L)}_n, h^{t,(L)}_{n+1}\}$. In particular, considering that learning tasks with different modes may need to extract different types of features, we add a learning task embedding vector w^{task} to the coding model of the converter to allow it to extract the information of a specific task at the time of output.

Based on the region embedding layer of the image and mark embedding layer of the text, the multilayer self-attention converter realizes the interaction of a single data stream. Each layer is composed of a self-attention mask mechanism and feedforward neural network. The structure of the Transformer encoder is the BERT model [16], which is highlighted in pink in Fig. 2, so the weight can be initialized with the pretrained BERT weight to improve the availability of the original pretraining model. This part uses the text and image encoders of the multilayer self-attention converter to encode the features of the problem text and image, respectively, so as to produce a better context feature representation.

3.1.1 Geometry Problem Text Feature Learning

Similar to the pretraining model BERT and its improved model [16–19] of natural language processing, the text input in the multimodal geometry problem is set as a set of word sequences $w = \{w_1, \cdots, w_n\}$, which is transformed to a set of text tag embedding sequences $W = \{E^t_{[CLS]}, E^t_{w_1}, \cdots, E^t_{w_n}, E^t_{[SEP]}\}$ through position, tag, and learning task embedding, where E and t are the text embedded tag and learning task embedded tag, respectively. The multilayer convertor, including a self-attention mask mechanism, is used to train the marked features of the problem text, learn the context marked representation of text features, and output the context marked representation. The learning task embedding vector w^{task}_t is added to the text tag embedding sequence as part of the BERT input, and can be separated from the implicit features of the output text $h^t = BERT(w, w^{task}_t)$ in the downstream task. This vector marks a variety of downstream training tasks and shares implicit features, such as geometry relation extraction, reasoning, and solution.

To better learn the context marked representation of text features, we use two types of language modeling tasks—bidirectional prediction and sequence-to-sequence generation—to train an encoding model of the problem text. The model uses a self-attention mask mechanism to control the context of prediction conditions. To improve the language learning process, we use the proposed syntactic-semantic model [4,6,33] to detect semantically complete phrases in the problem text. In the training process of bidirectional prediction [34] and sequence-to-sequence generation [32], we sample a sequence of complete words or phrases instead of word markers. In the whole training process, we evenly alternate training between bidirectional prediction and sequence-to-sequence generation targets, to obtain the context marked representation of text features.

3.1.2 Geometry Problem Image Feature Learning

Faster-RCNN [35] is used to detect the region of interest of the image to extract visual features. Because the self-attention mechanism in the multilayer converter is disordered, we use a five-dimensional vector, $(x_1/W,\ y_1/H,\ x_2/W,\ y_2/H,\ (y_2 - y_1)(x_2 - x_1)/WH)$, to encode the position features of each region, where (x_1, y_1) and (x_2, y_2) are the coordinates of the lower-left and upper-right corner, respectively, and W and H are the respective width and height of the input image. The region feature embedding sequence $V = \{E^i_{[IMG]}, E^i_{v_1}, \cdots, E^i_{v_m}\}$ is formed by using the first three layers of ResNet101 [22] to extract visual features, and a multilayer converter to fuse visual, location, and learning task features, where E is the image embedding mark of the region feature, and i is the embedding mark of the learning task. The multilayer converter including self-attention mask mechanism is used to train the region features, and to learn and output more realistic context region representation $h^i = \{h^{i,(L)}_0, h^{i,(L)}_1, \cdots, h^{i,(L)}_m\}$. The learning task embedding vector w^{task}_i is added to the region embedding sequence, and it can be separated from the implicit features of the output image $h^i = E(v, w^{task}_i)$ in the downstream task. This vector marks a variety of downstream training tasks and shares implicit features such as target detection, visual implication, and relational reasoning.

Similar to the self-attention mask modeling of text features, the self-attention mask modeling of image features samples the image region. It uses 15% probability to mask the visual features, which are replaced with 0. The image regions are usually highly overlapped. To avoid information leakage, we mask all high proportion of overlapping regions in the image slices. We randomly select a region as the masking anchor, and mask the region where the anchor receipt is greater than 0.3. The purpose is to reconstruct the masked region v_m from the other given region $v_{\setminus m}$ by using the function $L_V(\theta) = E_{v\in D}f_\theta(v_m|v_{\setminus m})$, where θ is a trainable parameter, and each image region v is sampled from training set D. Unlike text tags represented by discrete tags, visual features are high-dimensional and continuous, and the likelihood function cannot be used for supervised learning. Therefore, we use feature regression and region classification to train an encoding model of the geometry problem image, so as to learn a better context region representation of the image.

The process of forwarding propagation of data and the process of predicting the solution sequences are described in detail next. The model can accept single-modal and multimodal data as input. Data is converted to features by text and/or image embedders. As shown in algorithm 1, the input is the text and/or image and the output is a set of text features h^t and/or image feature h^i.

Algorithm 1: Multimodal feature extraction

Input: Word sequence $w = \{w_1, \cdots, w_n\}$ and/or problem image V, learning task embedding vector w_t^{task}

Output: Extracted text features h^t and/or image features h^i

```
1:  if SingleModalInput() then
2:      rewrite the semantics of single-modal text or single-modal image
3:  else
4:      rewrite original image and text pair data
5:  end if
6:  if TextExist() then
7:      add '[CLS]' to the beginning of sequences $w = \{[CLS], w_1, \cdots, w_n\}$
8:      add '[SEP]' to the end of sequences $w = \{[CLS], w_1, \cdots, w_n, [SEP]\}$
9:      put $w$ and $w_t^{task}$ into BERT to get $h^t = BERT\left(w, w_t^{task}\right)$
10:     get text tag embedding sequences $W = \left\{E^t_{[CLS]}, E^t_{w_1}, \cdots, E^t_{w_n}, E^t_{[SEP]}\right\}$ from $h^t$
11:     get $h^t = \left\{h_0^{t,(L)}, h_1^{t,(L)}, \cdots, h_n^{t,(L)}, h_{n+1}^{t,(L)}\right\}$ by putting $W$ into the multilayer self-attention converter
12: end if
13: if ImageExist() then
14:     image standardization pre-processing on $V$
15:     feed $V$ into pre-trained first three layers of ResNet101
16:     get image embedding $h^D$ from output of ResNet101
17:     put $h^D$ and $w_t^{task}$ into multilayer converter to get image feature $V = \left\{E^i_{[IMG]}, E^i_{v_1}, \cdots, E^i_{v_m}\right\}$.
18:     get $h^i = \left\{h_0^{i,(L)}, h_1^{i,(L)}, \cdots, h_m^{i,(L)}\right\}$ by putting $V$ into the multilayer self-attention converter
19: end if
20: return $h^t$ and/or $h^i$
```

3.2 Multimodal Contrastive Learning

We propose a multimodal contrastive learning model to address the lack of semantic correlation between multimodal geometry data. This model aligns different levels of text and/or image representations and maps them into a unified semantic space. The idea is to attract the relevant text and/or image features to each other and repel irrelevant features in the representation space [31], so as to realize the semantic alignment of multimodal geometry data.

The contrastive learning model of multimodal data is shown in Fig. 3. We transform the feature representation of a text and/or image into the same embedded space through a full connection layer, and calculate the cosine similarity to measure the distance $d(V, W)$ between features. Two text and/or image features, such as $h_m^{i,(L)}$ and $h_n^{t,(L)}$, are selected randomly, and two pairs of interrelated features are generated for each selected feature using two independent enhancement functions. A basic neural network encoder $f(\cdot)$ extracts representation vectors $(r_m^{i,(L)}, r_{m+1}^{i,(L)})$ and $(r_n^{t,(L)}, r_{n+1}^{t,(L)})$ from the enhanced data. A neural network projection head $g(\cdot)$ is used to map the representation vector to the contrastive loss space to obtain contrastive vectors $(z_m^{i,(L)}, z_{m+1}^{i,(L)})$ and $(z_n^{t,(L)}, z_{n+1}^{t,(L)})$. A contrastive loss function is trained for contrastive prediction, so that relevant features attract each other and irrelevant features repel each other.

3.2.1 Single-Modal Contrastive Learning

There is a lack of semantic correlation between the geometry data in a single-modal text or image. To enhance the multiple granularity semantic alignment between single-modal geometry data, we rewrite the semantics of a single-modal text or image using text rewriting technology. We parse the text or image into a scene graph [36] containing objects, attributes, and relations, which are randomly replaced in the geometric vocabulary. For instance, a geometry problem can be parsed into a scene graph of objects (circle, center, radius), attributes (center coordinate is (0, 0), radius is 2 cm) and relations (same radius). Text rewriting can be randomly replaced with other objects, attributes, or relations to generate a large number of negative samples. Because the rewritten text is similar to but different from the original text, it can be used as negative samples. Text rewriting can generate a large number of negative samples instead of randomly extracting negative samples, as in previous methods, so we can help the model learn a more detailed semantic alignment from different levels of a text or image, and a more accurate semantic alignment of features is learned by training the contrastive loss function,

$$L_{SMCL} = -\log \frac{\exp(d(Z, W^{+})/\tau)}{\sum_{W' \in \{W^{+}, W'\}} \exp(d(Z, W')/\tau)}, \tag{1}$$

where W^{+} and W^{-} represent positive and negative examples, respectively, of text or image Z, and τ represents temperature parameters. so as to realize the unified semantic feature expression of multimodal data.

3.2.2 Cross-Modal Contrastive Learning

For semantic alignment between cross-modal data in geometry problem texts and images, it is not only necessary to connect the scene displayed in the image with the text description in the problem, but also to align the entities in the image and their positional relationships with the description in the text. Many multimodal pretraining methods align visual and text representations by simple image and text matching using a corpus of restricted image and text pairs [23], randomly selecting negative samples of images or texts from the same training batch, and use a classifier to judge whether images and texts match. Because randomly selected negative samples are usually very different from the original images or texts, they can only learn a very rough alignment between text and visual representations. We use text rewriting to create a large number of positive and negative examples from the original image and text pair data instead of randomly extracting negative samples. We translate the description text of an image into another language, and use reverse translation technology [37] to translate it back to the original language, so as to obtain multiple similar description texts of images as positive samples. We retrieve the similarity of image descriptions through TF-IDF technology; the retrieval results are similar to but different from the original description text, so they can be used as negative samples to enhance the semantic alignment of image and text. We use these positive and negative examples to train the contrastive loss function,

$$L_{CMCL} = -\log \frac{\exp(d(V, W^{+})/\tau)}{\sum_{W' \in \{W^{+}, W'\}} \exp(d(V, W')/\tau)}, \tag{2}$$

where W^{+} and W^{-} represent positive and negative examples, respectively, of image V, and τ is a temperature parameter, to learn a more accurate semantic alignment between image and text representation, and realize the unified semantic feature expression of multimodal data.

The process of multimodal contrastive learning is described in detail next. As shown in algorithm 2, the input is the text and/or image encoded features and the output is a set of contrastive vectors $(z_{m}^{i,(L)}, z_{m+1}^{i,(L)})$ and $(z_{n}^{t,(L)}, z_{n+1}^{t,(L)})$.

Algorithm 2: Multimodal contrastive learning

```
Input:  Encoded text and/or image features $F_P$
Output: Contrastive vectors $(z_m^{i,(L)}, z_{m+1}^{i,(L)})$ and/or $(z_n^{t,(L)}, z_{n+1}^{t,(L)})$
1: randomly select encoded text and/or image features as $h_n^{t,(L)}$ and/or $h_m^{i,(L)}$ from $F_P$
2: use two independent enhancement functions to generate two pairs of interrelated features
3: use neural network encoder $f(\cdot)$ to extract representation vector $(r_m^{i,(L)}, r_{m+1}^{i,(L)})$ and/or $(r_n^{t,(L)}, r_{n+1}^{t,(L)})$
4: if SingleModalInput() then
5:     set contrastive loss function as $L_{SMCL}$
6: else
7:     set contrastive loss function as $L_{CMCL}$
8: end if
9: use projection head $g(\cdot)$ to map representation vector to the contrastive loss space
10: train the overall model by the contrastive loss function
11: obtain the contrastive vectors $(z_m^{i,(L)}, z_{m+1}^{i,(L)})$ and/or $(z_n^{t,(L)}, z_{n+1}^{t,(L)})$
12: return contrastive vectors $(z_m^{i,(L)}, z_{m+1}^{i,(L)})$ and/or $(z_n^{t,(L)}, z_{n+1}^{t,(L)})$
```

3.3 Geometry Problem Solver

3.3.1 Shared Encoder

We have multimodal information about images and text features or single-modal information to be encoded. Self-attention (SA) units are used to encode both embedded representations of text and images [38]. In the encoder, we concatenate the two kinds of feature vectors (h^{t} and h^{i}) for multimodal features, and feed single-modal features or concatenated multimodal features into six stacked SA units and obtain F_P, which are self-attended features.

Self-Attention Units. SA units are used to reconstruct every feature vector by attending all feature vectors. Multi-head is used to enhance generalization. We stack six SA units to reconstruct text-embedded feature h^{t} and image-embedded feature h^{i}. Each internal SA unit takes the output of the previous unit as input. SA units randomly mask h_j from h^t or h^i, and let the encoder explore deeper information from both feature vectors. It is worth noting that the masked feature vectors are not attended. We take the hidden state $F_P = [f_0; \ldots; f_n]$ of the last unit of SA as self-attended features. Most important is sharing parameters when encoding multimodal or single-modal information. Every output feature vector fully considers the entire feature information.

Adaptive Mechanisms. SA units take both text and image embedding as input, whose multimodal or single-modal embedding vector dimensions generally differ. SA can adaptively process vectors with different dimensions, providing the premise to construct a cross-modal information encoder and map them into a unified semantic space.

After features are extracted from the text and/or images, they are encoded for multimodal contrastive learning to map them into a unified semantic space. As described in Algorithm 3, the input is extracted text features h^t and/or image features h^i and the output is encoding result F_P.

Algorithm 3: Encoding of text and/or image features

```
Input:  Extracted text features $h^t$ and/or image features $h^i$
Output: Encoded text features and/or image features $F_P$
1: random masking of 15% of features as $H_p'$ and/or $h_D'$
2: if MultimodalFeatureExist() then
3:     concatenate $H_p'$ and $h_D'$ as $H'$
4: else
5:     treat $H_p'$ or $h_D'$ as $H'$
6: end if
7: feed $H'$ into 6 stacked SA units
8: get encoding $F_P = [f_0; \ldots; f_n]$ from output of last SA units
9: return $F_P$
```

3.3.2 Shared Decoder

The decoder part has one more cross-attention than SA, which is used to migrate cross-modal features into the shared features of multi-tasks. We stack six of the same structures in the decoder to represent deep information. It takes tokens of embedded learning tasks as input and pays attention to the cross-modal features from the encoder output. The list of learning tasks is manually marked to include all covered tasks. This paper focuses on problem solving in task-specific list. Cross-modal information from shared encoder output is used for cross attention, and it guides the decoder to generate task-specific solution information. Because text and image information is mapped to a unified semantic space, decoders are shared for different modal tasks.

3.3.3 Answer Generator

After obtaining the task-specific features of decoder output, we use a GRU network to generate a target operation tree with attention [39] over F_p. Operation trees are generated by preorder or postorder traversal for different datasets. Operation trees ensure the validity of the generated sequences, generating the current output node to identify the number of child nodes. For example, if the current node is "add," it has two child nodes, while a current node with a number has no child nodes. At the beginning of target program sequence, the tokens <CLS> and <END> are added to the beginning and end respectively of the sequence. We feed a start token <CLS> concatenated with decoded information into the GRU. For a target program sequence to predict $\{y_t\}$ $(1 \leq t \leq T)$, each prediction step is a classification issue. We can apply the validity of a generated tree to narrow the decision range. For example, in the subsequent tree, if two numbers have been generated, then the next generation must be the operation symbols, and we use this mechanism to improve the performance of the answer generator.

At the training state, after feeding the token <CLS>, we feed GRU hidden state s_0 concatenated with decoded information from the shared decoder to a linear layer with a softmax function to produce the probability of predicted next target program token distribution D_t. We feed golden target program token y_t concatenated with decoded information at time step t to produce the next prediction. At the final step, we feed the last program token concatenated with decoded information, and train the network to predict the <END> token, which means the predicted process is over. In the training process, the loss of answer sequences is the negative log-likelihood of the generated sequence

$$\mathcal{L}_a(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} \log P_t\left(y_t \mid \boldsymbol{x}, y_1, \ldots, y_{t-1}; \boldsymbol{\theta}\right), \tag{3}$$

where θ indicates parameters of the overall model structure without an image embedder, which is trained in auxiliary tasks. x is the input of both the problem text and image. $y_1, \ldots, y_{t-1}$ in training is the target sequence before time step t. For the test mode, it produces sequences in the model. Task-specific output heads can be customized for different tasks. Our answer generator has a geometry program generator.

We found that the model overfit quickly and performance was not increased in the early epoch. Flooding [40] was used to let the model take random walks at the empirical flooding level, which is the training loss of best performance of the model. We change the loss function to

$$\mathcal{L}'_a(\boldsymbol{\theta}) = \left|\mathcal{L}_a(\boldsymbol{\theta}) - b\right| + b, \tag{4}$$

where b is the training loss of the best performance of the model. The model will take gradient decrease as normal when training loss is greater than b, and gradient increase if it is less than b.

Algorithm 4 is built for decoding cross-modal information and generating solution sequences. After contrastive learning, the procedure takes the contrastive features z^t and/or z^i as the input and outputs the set of solution sequence $\mathcal{R}$. The contrastive features are first sent to the decoder, and then the solution sequence is generated by the answer generator.

4 Experiments

4.1 Experimental Setup and Implementation

4.1.1 Experimental Setup

We evaluate the multimodal contrastive learning capability of the proposed approach on the datasets of GeoQA [4] which contain 5010 Chinese geometry problems from online education websites. GeoQA contains angle, length, and other types of problems, as described in Table 1. Each problem contains an image and problem text with corresponding problem-solving explanations and an annotated program. The program can be used to model training and generate target sequences. To test the cross-modal contrastive learning ability of the model, it was simultaneously trained on the GeoQA and Math23K datasets [5]. Math23K was collected by Wang et al., and contains 23,162 tagged math word problems (MWPs), which are linear algebra questions with one unknown variable.

We used PyTorch to implement our model. The learning rate was set to $1e^{-3}$, with a batch size of 32 and 100 epochs with the Adam optimizer. The beam size was set to 10.

Algorithm 4: Generating targeted solution sequence

```
Input:  Contrastive features $z^t$ and/or $z^i$
Output: Solution sequence $\mathcal{R}$
1:  input embedding of geometry problem solving into decoder
2:  decoder cross-attention on $z^t$ and/or $z^i$
3:  get decoder output $F'$
4:  feed $F'$ into answer generator
5:  while predicted markers $\neq$ <END> do
6:      if IsFirstStep() then
7:          feed <CLS> concatenated with $F'$ into generator
8:      else
9:          take predicted target program token $y_{t-1}$ concatenated with $F'$ feed into generator
10:     end if
11:     feed hidden state of generator $s$ concatenated with $F'$ into a linear layer with a softmax
12:     get predicted target program $y_t$ from linear layer and add it to $\mathcal{R}$
13: end while
14: return $\mathcal{R}$
```

Table 1: Statistics of GeoQA and Math23K datasets

Dataset	GeoQA				Math23K
Number	Total	Angle	Length	Other	Total
	5010	2745	1873	392	23162

4.1.2 Implementation Details

We measured the multimodal contrastive learning capability of the model. For a fair comparison, we changed the answer generator to LSTM; because LSTM has more parameters than GRU and is more complex, we wanted to test for a corresponding performance improvement. To measure beam size influence in target program sequence generation, we contrasted our model with different beam sizes, so as to measure the performance improvement *vs.* the amount of computation. During the multimodal fusion experiment, the model converged quickly to the optimal solution, and overfit quickly. To moderate overfitting, the flooding method [3] was used, which let the model take random walks at a certain training loss level.

The cross-modal contrastive learning ability of the model was measured. The model was trained simultaneously on both datasets, and we trained it on a single dataset to see the corresponding performance changes. To measure the importance of the image and answer programs, we used a subset of the datasets for training, discarding images in GeoQA and discarding the solution sequence in GeoQA and Math23K.

For the GeoQA dataset, we calculated the answer accuracy of angle calculation, length calculation, and others such as area and volume calculation. Total problem solving performance was measured without distinction of problem types. For Math23K, we only evaluated the total problem solving performance. The answer accuracy was calculated by executing the predicted sequences to obtain the result compared to the correct answer. The consistency of the solution sequence was measured. The consistency of the marker sequence and model prediction sequence indicated the inference ability of the model, and the gap between consistency and accuracy indicated some inference ability.

For GeoQA, we first resized the image to 224×224, which is easy to feed into ResNet101. We fine-tuned ResNet101 to better extract image features, and pretrained it by puzzle recovery and image element identification with a learning rate of $1e^{-5}$. In the experiment with cross-modal data, the model was trained by two datasets at the same time. In each epoch, we selected a batch from the dataset in sequential rotation to train the model. At the end of each epoch, the models were evaluated on the validation sets of the two datasets, and the accuracy evaluation metrics were summed to select the model that performed best on both datasets, which was the final training result.

4.2 Experimental Results

4.2.1 Multimodal Contrastive Learning Results

Table 2 compares the result of solving geometry problems with the different methods on the GeoQA datasets. NGS-Auxiliary, Seq2Prog, BERT2Prog, MCL, LSTM Flooding, and MCL Flooding, respectively, refer to the performance of the NGS-Auxiliary method, Sequence-to-Program model using a GRU encoder with an attention mechanism [15], BERT [16] encoder with an attention mechanism, our proposed method, LSTM used to replace the answer generator with flooding, and our approach with flooding. Bolded scores are highest, and "‡" marks the results reported by Chen et al. [10]. Our model overall outperforms NGS-Auxiliary, but has 0.5 percentage point lower accuracy in solving triangle-type questions. At our flooding level setting, model performance has not improved, as MCL Flooding outperforms LSTM Flooding. The performance of Seq2Prog and BERT2Prog is not satisfactory because diagram information is lacking.

Table 2: Comparison results of solving geometry problems on GeoQA dataset

Method	Total (%)	Angle (%)	Length (%)	Other (%)
NGS-auxiliary‡ [4]	60.7	**72.0**	47.0	44.4
Seq2Prog‡ [15]	52.3	62.4	42.1	27.8
BERT2Prog‡ [16]	54.7	65.8	42.1	35.2
MCL (Ours)	**61.8**	71.5	**49.5**	**50.0**
LSTM flooding (Ours)	57.5	68.7	44.2	38.9
MCL flooding (Ours)	59.4	70.3	46.3	40.7

Fig. 4 shows the training and validation loss of different models. At the top-left corner, our validation loss begins to increase at the twelfth epoch. Although training loss always decreases, we achieve the best validation accuracy at the 42nd epoch, at 61.7%. We consider that the model is overfitting or the loss function must be improved. So, we try to change the answer generator to LSTM.

The graph titled "Replaced by LSTM" (lower-left) shows the LSTM-based answer generator validation loss increase at the twelfth epoch, with the best performance, 63.7% validation accuracy, at the 35th epoch. The two models show almost the same training trends, but the LSTM-based model performs best earlier. LSTM-based model learning curves are smoother than GRU-based, because LSTM is less likely to overfit, with more parameters than GRU. Flooding is used to overcome our model's tendency to overfit quickly. The training loss of the best-performing model on the validation set is used as the flooding level. For our model, the flooding level is set at 0.0248. For the LSTM-based answer generator, the flooding level is 0.1005. As shown in Table 2, our flooding level settings do not affect the accuracy improvement of the model. However, as seen in the graph titled "Our model with Flooding" (upper-right) in Fig. 4, flooding does alleviate a certain degree of overfitting, slowing down the upward trend of the validation set loss compared to the trend in the upper-left graph. The graph titled "Replaced by LSTM with Flooding" (lower-right) shows that for the LSTM-based answer generator, flooding is used to alleviate overfitting. The validation loss begins to increase at the sixteenth epoch. The best validation accuracy is achieved at 50 epochs with 0.624 accuracy.

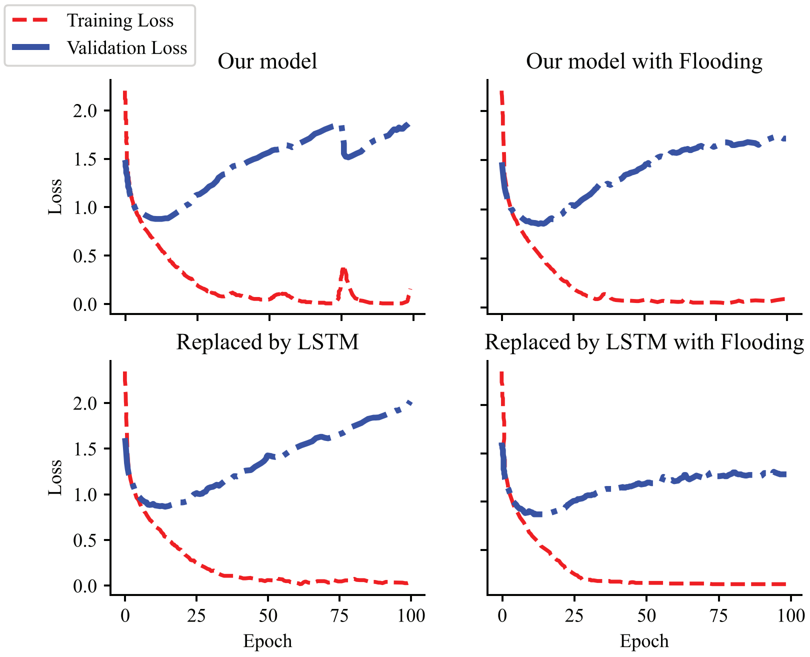


Figure 4: Training and validation loss of different models

The influence of different beam sizes was tested on our model. We set the beam size as 1, 10, and 20. As shown in Table 3, there is a huge accuracy gap between beam sizes 1 and 10. However, when the beam size increases from 10 to 20, the performance increase is not obvious.

Table 3: Performance comparison under different beam size settings of our model

Beam size	1	10	20
Answer accuracy	43.9%	61.8%	62.5%

4.2.2 *Cross-Modal Contrastive Learning Comparison Results*

Table 4 shows the answer accuracy of different settings on the two datasets which are GeoQA and Math23K, and "‡" marks the results reported by Wang et al. [41]. To compare model performance changes after cross-modal feature fusion, we used the proposed approach to train GeoQA and Math23K separately. "Only GeoQA," "Only Math23K," "Ensemble," "DNS," and "JMCL" refer to the performance of the implementation of only training GeoQA on our model, only training Math23K on our model, Wang's method with the same training set division of our approach, deep neural solver of MWP, and jointly training on the two datasets on our model. The result of JMCL is bolded.

Table 4: Comparison results of cross-modal fusion

Method	Datasets				
	GeoQA				Math23K
	Total (%)	Angle (%)	Length (%)	Other (%)	Total (%)
Only GeoQA (Multimodal)	60.7	72.0	47.0	44.4	-
Only Math23K (Single-modal)	-	-	-	-	67.6
Ensemble ‡ [41]	-	-	-	-	68.4
DNS‡ [42]					60.7
JMCL (Cross-modal)	**55.8**	**66.7**	**41.7**	**42.6**	**71.7**

We found performance degradation after fusion training for GeoQA, but there is some performance improvement for Math23K. Before adopting cross-modal contrastive learning, our model behaves at 67.6 accuracy on Math23K. After cross-modal contrastive learning, accuracy rises to 71.7, yielding a 4.1 percentage point increase, which is better than Wang's method Ensemble.

Table 5 shows the performance of using different data subsets to solve the problem. "No Image" and "No Program" refer to not using the image information in GeoQA and not using the program sequence of the two datasets.

Table 5: Answer accuracy comparison on different subsets of GeoQA and Math23K datasets

Method	Datasets				
	GeoQA				Math23K
	Total (%)	Angle (%)	Length (%)	Other (%)	Total (%)
No Image	**57.4**	**69.3**	**42.4**	**42.6**	**77.1**
No Program	26.7	26.1	26.5	27.8	14.7

No image information from GeoQA is used to solve the problem, so the model only processes single-modal information from both datasets. The first line, "No Image," shows that the accuracy is higher than JMCL, as Table 4 confirms, because unimodal information is more effective in contributing to each other's level of problem comprehension, so there is still much room for improvement in our cross-modal processing. The performance of "No Image" on the GeoQA dataset in Table 5 is less than that of "Only GeoQA" in Table 4 because of the variability between datasets. "No Image" has the best performance, 77.1, on Math23K. "No Program" in Table 5 means the answer is chosen directly from the four options without undertaking any solving steps. The second line, "No Program," shows that there is a significant drop in performance and the model fails to reason about problems and prove the importance of program sequence.

The consistency and Levenshtein distance of the solution sequence were measured. As Table 6 shows, our cross-modal contrastive learning model has 37.7 consistency on the GeoQA dataset, which is 18.1 different from accuracy, and for Math23K, there is a gap of 9.7. We suggest that the gap occurs because the model is able to generate different solution steps from the labeled sequence to solve the problem. Because a problem does not have a unique solution, the ability to generate different sequences shows some reasoning ability. For a more comprehensive analysis, the Levenshtein distance, a string metric, was measured; this measures the difference between two sequences. The Levenshtein distance of the predicted and labeled sequences was better than the consistency because it only shows the similarity of the sequences. Levenshtein also shows the performance of the model. As Table 6 shows, the Levenshtein distance of the model when trained with Math23K only was 85.0, but it was 86.6 when fusion training of the two datasets was performed. For GeoQA dataset performance, there was no such change.

Table 6: Consistency and Levenshtein distance of solution sequence obtained by our model

Method	Dataset			
	GeoQA		Math23K	
	Consistency (%)	Levenshtein (%)	Consistency (%)	Levenshtein (%)
Only GeoQA (Multimodal)	**51.1**	**82.5**	-	-
Only Math23K (Single-modal)	-	-	57.7	85.0
JMCL (Cross-modal)	37.7	75.3	62.0	86.6
No Image	38.5	74.5	**66.5**	**88.5**

To explain the performance degradation of GeoQA, we speculate that Math23K has much more data, which increases its impact on the model, and the organization rules of the two datasets labeled with sequences are different. Fig. 5 illustrates that GeoQA and Math23K have different forms of solution sequence construction. In GeoQA, operation symbols include “Minus” and “Half”, and the operands include “N0” and “V1”. For Math23K, operation symbols include “-” and “/”, and operand include “temp_b” and “temp_c”. In the two datasets, different symbols are used for the same operation, making the vocabularies of the model poorly generalized. GeoQA uses preorder trees for operations, while Math23K uses postorder trees. For these reasons, the two datasets do not effectively contribute together to the model’s understanding of the problem.

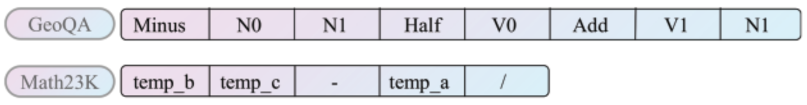
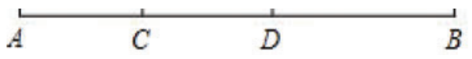


Figure 5: Examples of solution sequences for GeoQA and Math23K

4.3 Typical Case Analyzing

Two typical cases have been displayed below. Typically, if the predicted program sequence has no match to any of the options, then the model gets no result and we do not let the model make random choices. In Figs. 6 and 7, the operation is predefined such as Minus and Add. Those operations will be executed by pre-defined functions to get a result and the result of each operation is stored in a variable to facilitate the final result. It is worth noting that an operation is generated firstly and secondly, the numbers involved in the operation are generated in annotated program sequences. In programs, N means number in the problem. The serial number is the order in which the numbers appear. V means variables that are operation results labeled by order of execution. C is a constant that has been predefined. The final feature representations of problem are fed into a classifier which transforms features into probability distribution. The position of the maximum value in distribution corresponds to the predicted program. Each time step of classifier will produce a program such as "Minus" until overall program sequences are generated.

As shown in the figure, C and D are two points on the line AB, if AC = 3cm, C is the midpoint of AD and AB = 10cm, then DB=( ).



A.4cm B.5cm C.6cm D.7cm

Answer: A.0

Explanations:

∵ Point C is the midpoint of AD, AC=3cm,

∴ CD=3cm.

∵ AB=10cm, AC+CD+DB=AB,

∴ BD=10-3-3=4cm.

Therefore choose: A

Annotated Program:

Minus	N1	N0	Minus	V0	N0

Model Predicted:

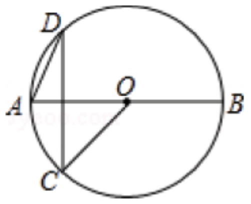
Minus	N0	N1	Half	V0	Add	V1	N1

Figure 6: Case of our model gets no result

In Fig. 6, the model gets the wrong answer sequence and the result is 6.5. At the beginning of the sequence, the model predicts $N_0 - N_1$ which is $3 - 10 = -7$, and -7 is stored in the variable V_0. Then the model predicts "Half" and the operation number are V_0 and V_1 which obtained $(-3.5 = -7/2)$. Finally, the answer has been got by $Answer = V1 + N1(6.5 = -3.5 + 10)$. We find that in length type questions neither the final result nor the intermediate result will be negative. In future works, we try to avoid negative forecasts and improve the performance of the model by avoiding operations that can generate negative numbers. The model may have failed to effectively perform Multimodal information fusion.

In Fig. 7, the model predicts the right target programs. C_2 and C_3 are constants that are defined in advance. In our setting, C_2 is 90 and C_3 is 180. We can get $V_0 = C_2 - N_0 = 90 - 70 = 20$, then $V_1 = V_0 * 2 = 40$. Finally, we get $Answer = C_3 - V_1 = 140$. Interestingly, the model can use predefined constants to solve the problem, verifying the validity of the model and demonstrating powerful inference capability.

As shown in the figure, AB is the diameter of ⊙O, C and D are two points on ⊙O, CD⊥AB, if ∠DAB = 70°, then ∠BOC=( ).



A.70.0 B.130.0 C.140.0 D.160.0

Answer: B.140.0

Explanations:

∵CD⊥AB. ∠DAB=70°,

∴∠ADC=90°-∠DAB=20°,

∴∠AOC=2∠ADC=40°,

∴∠BOC=180°-∠AOC=140°.

Therefore choose: C

Annotated Program:

Minus	C2	N0	Double	V0	Minus	C3	V1

Model Predicted:

Minus	C2	N0	Double	V0	Minus	C3	V1

Figure 7: Case of our model gets right answer

5 Conclusion

In this work, we focused on solving geometry problems via the feature and contrastive learning of multimodal data. A shared feature-learning model of multimodal data was adopted to learn a unified feature representation of a text and image in order to address the heterogeneity between multimodal geometry problems. A contrastive learning model of multimodal data was proposed to enhance the semantic relevance between multimodal features and map them into a unified semantic space. This model can effectively adapt to both single-modal and multimodal downstream tasks. A shared encoder-decoder structure realized the semantic alignment of multimodal geometry data and generated readable solving sequences of problems. The shared encoder processed text and/or image features masked by self-attention units, and multilayer Transformer was used to realize the interaction between cross-modal features. After encoding, multimodal contrastive learning was proposed to realize the semantic alignment of multimodal geometry data. The shared decoder processed contrastive features and used learning task lists to transform information to generate task-specific features. The method can be used for a variety of applications without changing too much structure. The experimental results showed that the proposed method is promising in the solution of geometry problems.

In future work, we will eliminate the variability in the solution sequences of different datasets to improve the performance of the model, and explore a higher-performance solution framework through methods such as reinforcement learning.

Acknowledgement: The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation. We also thank LetPub (www.letpub.com) for its linguistic assistance during the preparation of this manuscript.

Funding Statement: This work is supported by the National Natural Science Foundation of China (No. 62107014, Jian P., 62177025, He B.), the Key R&D and Promotion Projects of Henan Province (No. 212102210147, Jian P.), and Innovative Education Program for Graduate Students at North China University of Water Resources and Electric Power, China (No. YK-2021-99, Guo F.).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Zhang, J. (2000). *Mathematics mechanization and applications.* Cambridge: Academic Press.
2. Zhang, D., Wang, L., Zhang, L., Dai, B. T., Shen, H. T. (2019). The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(9),* 2287–2305. DOI 10.1109/TPAMI.34.
3. Sachan, M., Dubey, A., Hovy, E. H., Mitchell, T. M., Roth, D. et al. (2020). Discourse in multimedia: A case study in extracting geometry knowledge from textbooks. *Computational Linguistics, 45(4),* 627–665. DOI 10.1162/coli_a_00360.
4. Yu, X., Wang, M., Gan, W., He, B., Ye, N. (2019). A framework for solving explicit arithmetic word problems and proving plane geometry theorems. *International Journal of Pattern Recognition and Artificial Intelligence, 33(7),* 1940005. DOI 10.1142/S0218001419400056.
5. Gan, W., Yu, X., Wang, M. (2019). Automatic understanding and formalization of plane geometry proving problems in natural language: A supervised approach. *International Journal on Artificial Intelligence Tools, 28(4),* 1940003. DOI 10.1142/S0218213019400037.
6. Gan, W., Yu, X., Zhang, T., Wang, M. (2019). Automatically proving plane geometry theorems stated by text and diagram. *International Journal of Pattern Recognition and Artificial Intelligence, 33(7),* 1940003. DOI 10.1142/S0218001419400032.
7. Seo, M., Hajishirzi, H., Farhadi, A., Etzioni, O., Malcolm, C. (2015). Solving geometry problems: Combining text and diagram interpretation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1466–1476. Lisbon.
8. Seo, M. J., Hajishirzi, H., Farhadi, A., Etzioni, O. (2014). Diagram understanding in geometry questions. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2831–2838. Quebec.
9. Lu, P., Gong, R., Jiang, S., Qiu, L., Huang, S. et al. (2021). Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. *arXiv preprint arXiv:2105.04165*.
10. Chen, J., Tang, J., Qin, J., Liang, X., Liu, L. et al. (2021). GeoQA: A geometric question answering benchmark towards multimodal numerical reasoning. *arXiv preprint arXiv:2105.14517*.
11. Zhong, X., Fu, H., Yu, Y., Liu, Y. (2015). Interactive learning environment based on knowledge network of geometry problems. *10th International Conference on Computer Science & Education (ICCSE)*, pp. 53–58. Cambridge.

12. Wang, L., Zhang, D., Gao, L., Song, J., Guo, L. et al. (2018). Mathdqn: Solving arithmetic word problems via deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5545–5552. New Orleans.
13. Nam, H., Ha, J. W., Kim, J. (2017). Dual attention networks for multimodal reasoning and matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 299–307. Hawaii.
14. Lu, J., Yang, J., Batra, D., Parikh, D. (2016). Hierarchical question-image co-attention for visual question answering. *Advances in Neural Information Processing Systems, 29,* 289–297.
15. Amini, A., Gabriel, S., Lin, P., Koncel-Kedziorski, R., Choi, Y. et al. (2019). MathQA: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.
16. Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
17. Dong, L., Yang, N., Wang, W., Wei, F., Liu, X. et al. (2019). Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems, 32,* 12727–12739.
18. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M. et al. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
19. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. et al. (2019). XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems, 32,* 5571–5581.
20. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A. et al. (2019). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.
21. Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
22. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778. Las Vegas.
23. Chen, Y. C., Li, L., Yu, L., El Kholy, A., Ahmed, F. et al. (2020). Uniter: Universal image-text representation learning. *European Conference on Computer Vision*, Glasgow, Scotland, pp. 104–120.
24. Li, L. H., Yatskar, M., Yin, D., Hsieh, C. J., Chang, K. W. (2019). VisualBERT: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.
25. Su, W., Zhu, X., Cao, Y., Li, B., Lu, L. et al. (2019). VL-BERT: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*.
26. Lu, J., Batra, D., Parikh, D., Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in Neural Information Processing Systems, 32,* 1–10.
27. Nie, L., Wang, W., Hong, R., Wang, M., Tian, Q. (2019). Multimodal dialog system: Generating responses via adaptive decoders. *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 1098–1106. France.
28. Nie, L., Jia, M., Song, X., Wu, G., Cheng, H. et al. (2021). Multimodal activation: Awakening dialog robots without wake words. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500. Canada.
29. Nie, L., Jiao, F., Wang, W., Wang, Y., Tian, Q. (2021). Conversational image search. *IEEE Transactions on Image Processing, 30,* 7732–7743. DOI 10.1109/TIP.2021.3108724.
30. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z. et al. (2018). AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1316–1324. Salt Lake City.

31. Zhang, H., Koh, J. Y., Baldridge, J., Lee, H., Yang, Y. (2021). Cross-modal contrastive learning for text-to-image generation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 833–842. Nashville.
32. Li, W., Gao, C., Niu, G., Xiao, X., Liu, H. et al. (2020). UNIMO: Towards unified-modal understanding and generation via cross-modal contrastive learning. *arXiv preprint arXiv:2012.15409*.
33. Jian, P., Sun, C., Yu, X., He, B., Xia, M. (2019). An end-to-end algorithm for solving circuit problems. *International Journal of Pattern Recognition and Artificial Intelligence, 33(7),* 1940004. DOI 10.1142/S0218001419400044.
34. Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L. et al. (2020). Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics, 8,* 64–77. DOI 10.1162/tacl_a_00300.
35. Ren, S., He, K., Girshick, R., Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems, 28,* 91–99.
36. Wang, Y. S., Liu, C., Zeng, X., Yuille, A. (2018). Scene graph parsing as dependency parsing. *arXiv preprint arXiv:1803.09189*.
37. Edunov, S., Ott, M., Auli, M., Grangier, D. (2018). Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.
38. Yu, Z., Yu, J., Cui, Y., Tao, D., Tian, Q. (2019). Deep modular co-attention networks for visual question answering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6281–6290. California.
39. Bahdanau, D., Cho, K., Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
40. Ishida, T., Yamane, I., Sakai, T., Niu, G., Sugiyama, M. (2020). Do we need zero training loss after achieving zero training error?. *arXiv preprint arXiv:2002.08709*.
41. Wang, L., Wang, Y., Cai, D., Zhang, D., Liu, X. (2018). Translating a math word problem to an expression tree. *arXiv preprint arXiv:1811.05632*.
42. Wang, Y., Liu, X., Shi, S. (2017). Deep neural solver for math word problems. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 845–854. Copenhagen.