Template-Based Contrastive Distillation Pretraining for Math Word Problem Solving

Jinghui Qin, Zhicheng Yang, Jiaqi Chen, Xiaodan Liang, and Liang Lin, *Senior Member, IEEE*

***Abstract*— Since math word problem (MWP) solving aims to transform natural language problem description into executable solution equations, an MWP solver needs to not only comprehend the real-world narrative described in the problem text but also identify the relationships among the quantifiers and variables implied in the problem and maps them into a reasonable solution equation logic. Recently, although deep learning models have made great progress in MWPs, they ignore the grounding equation logic implied by the problem text. Besides, as we all know, pretrained language models (PLM) have a wealth of knowledge and high-quality semantic representations, which may help solve MWPs, but they have not been explored in the MWP-solving task. To harvest the equation logic and real-world knowledge, we propose a template-based contrastive distillation pretraining (TCDP) approach based on a PLM-based encoder to incorporate mathematical logic knowledge by multiview contrastive learning while retaining rich real-world knowledge and high-quality semantic representation via knowledge distillation. We named the pretrained PLM-based encoder by our approach as MathEncoder. Specifically, the mathematical logic is first summarized by clustering the symbolic solution templates among MWPs and then injected into the deployed PLM-based encoder by conducting supervised contrastive learning based on the symbolic solution templates, which can represent the underlying solving logic in the problems. Meanwhile, the rich knowledge and high-quality semantic representation are retained by distilling them from a well-trained PLM-based teacher encoder into our MathEncoder. To validate the effectiveness of our pretrained MathEncoder, we construct a new solver named MathSolver by replacing the GRU-based encoder with our pretrained MathEncoder in GTS, which is a state-of-the-art MWP solver. The experimental results demonstrate that our method can carry a solver's understanding ability of MWPs to a new stage by outperforming existing state-of-the-art methods on two widely adopted benchmarks Math23K and CM17K. Code will be available at https://github.com/QinJinghui/tcdp.**

***Index Terms*— Contrastive learning, math word problem (MWP) solving automatically, model pretraining, natural language understanding.**

Manuscript received 11 April 2022; revised 10 July 2022, 28 December 2022, and 24 March 2023; accepted 1 April 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0109700, in part by the National Natural Science Foundation of China (NSFC) under Grant 62206314 and Grant 61976233, in part by the National Natural Science Foundation of Guangdong Province under Grant 2022A1515011835, and in part by the China Postdoctoral Science Foundation Funded Project under Grant 2021M703687. *(Jinghui Qin and Zhicheng Yang contributed equally to this work.) (Corresponding author: Xiaodan Liang.)*

Jinghui Qin, Jiaqi Chen, and Liang Lin are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China (e-mail: qinjingh@mail2.sysu.edu.cn; jadgechen@gmail.com; linliang@ieee.org).

Zhicheng Yang is with the School of Intelligent Systems Engineering, Shenzhen Campus, Sun Yat-sen University, Shenzhen, China, and also with Dark Matter Inc., Guangzhou 511457, China (e-mail: yangzhch6@mail2.sysu.edu.cn).

Xiaodan Liang is with the School of Intelligent Systems Engineering, Shenzhen Campus, Sun Yat-sen University, Shenzhen, China (e-mail: xdliang328@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3265173.

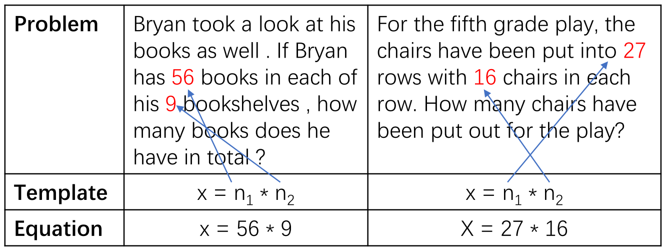
Digital Object Identifier 10.1109/TNNLS.2023.3265173

I. Introduction

SOLVING math word problem (MWP) is a long-standing challenging natural language processing (NLP) task and has been gaining more attention in the research community recently [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. A typical MWP is a short narrative that describes a real-world situation and inquires about one or more unknown facts in the situation. To solve an MWP, an MWP solver not only needs to comprehend the real-world narrative described in the problem text but also needs to possess mathematical reasoning skills to identify the relationships among the quantifiers and variables implied in the problem and map them into a reasonable solution equation logic [11], as shown in Fig. 1.

Recently, large-scale pretrained language models (PLMs), such as ELMo [12], GPT [13], BERT [14], and XLNet [15], have archived great advances on various NLP tasks benefiting from the powerful learned linguistic representation and knowledge representation based on a large corpus. The success of large-scale pretraining technology also drives the development of pretrained models in other fields, such as CodeBERT [16] for programming, BioBERT [17] for medicine, VideoBERT [18] for multimodal video understanding, and LayoutLM [19] for document image understanding.

Inspired by the above works, we believe that an MWP solver's ability of problem comprehension can be lifted by encoding problem text with the PLMs due to their entailed real-world knowledge and high-quality semantic representation. However, most existing PLMs are trained on the large-scale unsupervised corpus to learn general representation for general language understanding. They ignore task-specific knowledge, such as the solution logic implied by MWP text for solving an MWP, which makes them unable to fully understand MWPs. Therefore, enforcing pretraining again based on task-specific corpus is the key to giving full play to language modeling ability for comprehending MWPs. Besides, we also find that if different word problems with different natural language descriptions can be solved with the same symbolic solution expression, their solution logics are also the same. As shown in Fig. 1, two different word problems can be solved with the same logic, which is the product of the number of the first entity (books versus chairs) and the number of the



Problem	Bryan took a look at his books as well . If Bryan has 56 books in each of his 9 bookshelves , how many books does he have in total ?	For the fifth grade play, the chairs have been put into 27 rows with 16 chairs in each row. How many chairs have been put out for the play?
Template	x = $n_1 * n_2$	x = $n_1 * n_2$
Equation	x = 56 * 9	X = 27 * 16

Fig. 1. Two different MWPs with the same solving logic. Although the two problems describe different real-world scenes, they share the same solving logic: the first number multiply by the second number.

second entity (bookshelves versus rows) if we strip out specific language descriptions.

To harvest the equation logic and real-world knowledge simultaneously, we propose a template-based contrastive distillation pretraining (TCDP) approach to further pretrain PLM-based encoder to incorporate mathematical logic knowledge by multiview contrastive learning while retaining rich real-world knowledge and high-quality semantic representation via knowledge distillation. For convenience, we named the pretrained PLM-based encoder with our approach as MathEncoder. Specifically, we first introduce symbolic formula templates to summarize the underlying solving logic in MWPs. Then, the mathematical logic is captured via supervised contrastive learning based on expression templates, which can represent the underlying solving logic in the problems. However, injecting solving logic into problem representation directly will damage the common linguistic representation capability of the PLMs, which is also crucial for a solver to understand MWPs comprehensively. To address this issue, we further incorporate knowledge distillation at the token level into our pretraining procedure to retain rich real-world knowledge and well-trained semantic information by distilling features from a well-trained PLM-based teacher encoder into our MathEncoder.

To validate the effectiveness of our pretrained MathEncoder in the downstream MWP-solving task, we construct a new solver named MathSolver by replacing the GRU-based encoder with our pretrained MathEncoder in GTS [5], which is a state-of-the-art MWP solver. The experimental results demonstrate that our method can carry a solver's understanding ability of MWPs to a new stage by outperforming existing state-of-the-art methods on two widely adopted benchmarks Math23K and CM17K. Besides, to show the few-shot ability of our MathSolver, we further conduct experiments on different data usages, which show that our MathSolver can outperform other methods in the few-shot learning setting.

In summary, the main contributions of this work are threefold as follows.

1) To the best of our knowledge, we are the first study to introduce symbolic formula templates to summarize the underlying solving logic in MWPs and adapt it into our pretraining procedure.
2) We propose a novel TCDP approach on a PLM-based encoder to incorporate mathematical logic knowledge by multiview template-based contrastive pretraining as well as applying knowledge distillation to retain rich real-world knowledge and common linguistic information.
3) We construct a new solver named MathSolver based on our MathEncoder pretrained by our proposed TCDP approach to show the effectiveness and superiority of our approach on the downstream MWP-solving task. The experimental results demonstrate that our method can carry a solver's understanding ability of MWPs to a new stage on two widely adopted benchmarks Math23K and CM17K.

II. Related Works

A. Word Problem Solving

In recent years, deep learning-based models [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [20], [21], [22], [23], [24], [25], [26], [27] have shown impressive performance in solving MWPs by automatically learning to directly translate a problem text into an expression without any handcrafted feature design. All these methods follow the RNN-based encoder–decoder paradigm with some different designs. Wang et al. [1] made the first attempt to apply a vanilla sequence-to-sequence (seq2seq) model to translate the language text to a solution expression. Huang et al. [2] improved their work by introducing a copy and attention mechanism. Xie and Sun [5] proposed a tree-structured decoder to decode expressions in prefix order. Furthermore, Zhang et al. [8] improved problem text representation by fusing quantity-related graph encoder. Hong et al. [28] proposed to train a solver in a weakly supervised way by constructing pseudolabels during training. Hong et al. [29] also proposed a situation model for algebra story problems via attributed grammar. Qin et al. [10] proposed multiple auxiliary tasks to improve problem text representation and the ability to predict common-sense constants. Wu et al. [21] enhanced MWP-solving performance by explicitly incorporating numerical values into a sequence-to-tree network and applying a numerical properties prediction mechanism. Shen et al. [26] devised a new ranking task for MWP and proposed the Generate and Rank, a multi-task framework based on a generative PLM. Wu et al. [24] proposed a novel edge-enhanced hierarchical graph-to-tree model, in which the MWPs are represented as edge-labeled graphs. Liang and Zhang [20] designed a teacher module to make the MWP encoding vector match the correct solution and discard the wrong solutions. Cao et al. [22] proposed a novel Seq2DAG approach to extract equation set directly as a DAG structure. Lin et al. [23] proposed a hierarchical math solver to make deep understanding and exploitation of problems by imitating human reading habits. Huang et al. [25] proposed a novel human-like analogical learning method in a recall and learn manner to improve performance. Lee et al. [27] proposed template-based multitask generation to improve the problem-solving accuracy of the mathematical word problem-solving task by sufficiently utilizing numeric information. Xiong et al. [30] proposed a solver based on variational information bottleneck to extract essential features of the expression syntax tree while filtering latent-specific redundancy containing syntax-irrelevant features.

In this work, we propose a novel TCDP approach on a PLM-based encoder to incorporate mathematical logic knowledge by multiview template-based contrastive pretraining as well as applying knowledge distillation to retain rich real-world knowledge and common linguistic information.

B. Pretrained Language Models

Pretrained models, such as ELMo [12], GPT [13], BERT [14], and XLNet [15], have attracted wide attention, since they greatly improve the performance of various NLP tasks, such as text classification [14], machine translation [31], [32], and so on. However, these models were not trained on the in-domain corpus, which makes them hard to handle specific objects better. Thus, some specific pretrained models were proposed for specific objects. For example, VideoBERT [18] is a pretrained model for text-to-video generation and future forecasting by jointly training on video and natural languages. CodeBERT [16] is trained in programming and natural languages for code synthesis. BioBERT [17] is trained for biomedical language representation and biomedical text mining. LayoutLM [19] is trained for providing more informative representations for document image understanding.

Different from the above works, which usually adopted unsupervised pretraining, in this work, we pretrain an encoder for MWP solver with an in-domain dataset in the supervision way, so that we can capture solving logic for better MWP understanding.

C. Contrastive Representation Learning

Contrastive learning is a method of representation learning, which is first proposed in the article [33]. The core of contrastive learning is providing more effective representations by pulling semantically similar embeddings together and pushing semantic different ones apart. In recent, contrastive learning has been explored in many fields and made great progress in density estimation and representation learning, especially in self-supervised setting [34]. Contrastive learning was deployed to reduce word omission errors in neural machine translation [35]. A discriminative model was trained on contrastive examples to obtain more effective language representation [36]. Contrastive learning was used in supervised and unsupervised settings to advance the performance of the sentence embeddings [37]. Besides, it has been extended to knowledge distillation for image classification tasks [38].

Different from prior works, we propose supervised contrastive objectives based on expression templates for MWP understanding. With our objectives, we can pull solving logic-consistent MWPs together and push inconsistent MWPs apart, so that we can build more informative semantic representations for MWPs.

D. Knowledge Distillation

Since Hinton et al. [39] proposed to distill the knowledge of large models into smaller and faster models without losing too much generalization performance, knowledge distillation has become one of the standard ways for model compression [40], [41], [42], [43] by effectively learning a small student model from a large teacher model. The sequence-level knowledge distillation was proposed to improve the inference speed and performance of neural machine translation [44]. To transfer knowledge from an ensemble model to a single model for machine reading comprehension, knowledge distillation was applied to answer span prediction [45]. The structured distillation scheme [41] was proposed to distill the structured knowledge from large networks to small networks. Matching guided distillation (MGD) [46] was proposed to mitigate the gap in semantic feature structure between the intermediate features of teacher and student. Semantic calibration [47] for cross-layer knowledge distillation was proposed to automatically assign proper target layers of the teacher model for each student layer with attention. ICKD [48] was proposed to retain interchannel correlation of features to capture the intrinsic distribution of the feature space and sufficient diversity properties of features in the teacher network.

Different from prior works, we adopt knowledge distillation to retain semantic information maximally while conducting template-based contrastive pretraining for injecting solving logic into problem representations.

III. Preliminaries

A. MWP and Preprocessing

An MWP usually consists of a problem text P and a solution expression tree S, where the following hold.

1) *Problem Text P:* The problem text P consists of a sequence of word tokens and numeric values, which is often a short narrative that describes the changes of the world state and poses a question about an unknown quantity or multiple questions about multiple unknown quantities. Since the numeric values are varied and hard to include all different numeric values in the vocabulary, all the numeric values are treated as a special word token [NUM], so that we can represent various numeric values with a special token uniformly, and an MWP solver can focus on understanding or capturing the mathematical relationships among numeric values without considering their exact values. The special word token [NUM] is also added to the PLM-based encoders' vocabulary. The PLM-based encoder may be BERT [14] or Roberta [49]. To track the exact numeric values, we also map each numeric value in a problem P to a symbolic token list $\{n_1, n_2, n_3, \dots\}$ in the order that they appear in the problem text, so that we can recover the symbolic number tokens into the exact numeric values for obtaining a calculable solution expression.
2) *Expression Tree S:* The solution expression tree S is a mathematical expression tree transformed from the solution expression by using a recursive traversal algorithm. In general, the solution expression tree S consists of constant quantities, mathematical operators, and numeric values from the problem text P. In our definition, the solution expression only involves binary math operators $\{+, \text{-}, *, /, \wedge, =\}$; thus, the expression tree S is a binary tree. For those problems with multiple solution

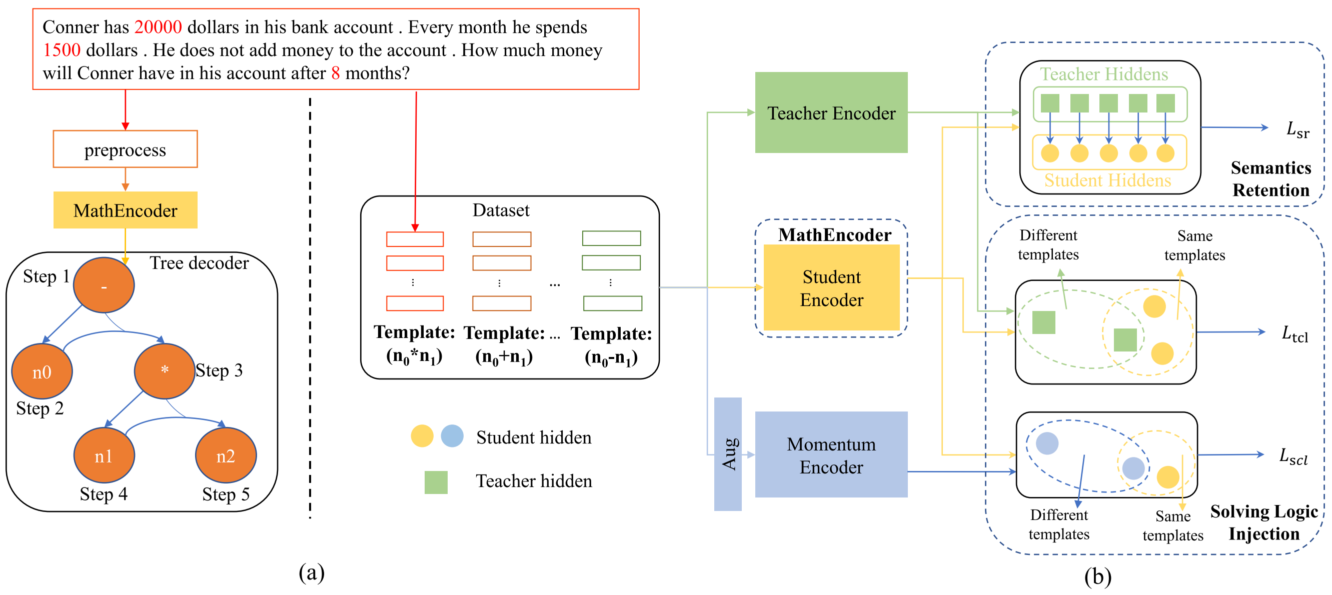


Fig. 2. Overview of our TCDP approach. (a) Structure of MathSolver in which our MathEncoder extracts high-quality semantic representation, and the tree decoder adapted from GTS [5] generates an expression tree according to the preorder traversal method. (b) Overview of our TCDP procedure. We pretrain our MathEncoder with three objectives for injecting solving logic and keeping original semantic information simultaneously. $\mathcal{L}_{\text{scl}}$ and $\mathcal{L}_{\text{tcl}}$ are used for solving logic injection, while the values of $\mathcal{L}_{\text{sr}}$ are adopted for semantic information retention. (The circle indicates student hidden vector, while the square indicates teacher hidden. The hidden vectors in the yellow dotted circle belong to the same expression template.)

TABLE I

STATISTICS OF EXPRESSION TEMPLATES IN DIFFERENT SUBSETS ON MATH23K AND CM17K

Dataset	Total	Training Set	Valid Set	Test Set
Math23K [1]	3790	3576	465	467
CM17K [10]	9918	8325	1404	1396

expressions, we also construct a universal expression tree with the universal expression tree representation [9]. Constant quantities consist of some specific values that may not appear in the problem text, such as $\{2, 3.14\}$. To decrease the diversity of expression trees, accelerate learning procedure, and generalize better, the numeric values will be mapped to $\{n_1, n_2, n_3, \ldots\}$ in the order they appear in the problem text P. As shown in Fig. 2(a), all the leaf nodes in an expression tree are constant quantities and numeric values, and nonleaf nodes are binary math operators.

B. Template Generation

As logic knowledge for pretraining, the expression template T is generated automatically by mapping all the numbers in solution expression, which is transformed from the expression tree S in infix traversal to $\{n_1, n_2, n_3, \ldots\}$ in the order they appear in the problem text P in preprocessing procedure. As a result, there are 3790 expression templates in Math23K [1], while there are 9918 expression templates in CM17K [10]. The statistics of expression templates in different subsets on Math23K and CM17K are shown in Table I. We also show the statistics of the commonly used templates in Math23K [1] dataset in Table II.

C. MWP Solving

To solve an MWP, an MWP solver needs to transfer a problem text P to a solution expression tree S, as shown in Fig. 2(a). Therefore, the task of MWP solving can be defined formally as follows.

TABLE II

COMMONLY USED TEMPLATES IN MATH23K

Templates	Count	Templates	Count
$x = n_0 * n_1$	1105	$x = n_0 * n_1 + n_2$	385
$x = n_0 / n_1$	1018	$x = n_0 * n_1 * n_2$	354
$x = n_1 / n_0$	794	$x = n_1 / (1 - n_0)$	331
$x = n_0 * (1 - n_1)$	536	$x = (n_0 - n_2) / n_1$	327
$x = n_0 * n_1 / n_2$	519	$x = n_0 / (1 - n_1)$	291

Given an MWP training dataset $\mathcal{D} = \{(P_i, S_i)\}_{i=1}^{N}$, where N is the size of the training set, we aim to learn a neural solver $f(S_i|P_i)$ that takes an MWP P_i as input and outputs the targeted symbolic expression S_i, where (P_i, S_i) is the labeled data pair. For mining effective information and achieving good performance, the main learning objective of a neural solver is to minimize the negative log-likelihood (NLL) loss function as follows:

$$\mathcal{L}_{\text{NLL}} = \frac{1}{N}\sum_{i=1}^{N} -\log f(S_i|P_i) \tag{1}$$

where $f(S_i|P_i)$ is the output distribution over sequences.

IV. TCDP

In this section, we first provide an overview of MathEncoder and MathSolver and then describe the details of our proposed TCDP method.

A. MathEncoder and MathSolver

An MWP solver should possess rich real-world knowledge and mathematical reasoning skills [11], the encoder–decoder architecture can better meet this condition in which

PLM-based encoder can be considered as a language comprehension module with rich real-world knowledge, and the tree decoder can be considered as an algebraic reasoning module.

Therefore, to construct a suitable backbone for MWP solving, we first deploy a PLM as an encoder due to its rich real-world knowledge, such as BERT [14], [50] and Roberta [49]. Then, we deploy a tree decoder adapted from GTS [5] as a decoder to generate expression tree nodes step by step in prefix order. As illustrated in Fig. 2(a), the left child node is generated according to its parent node. When it comes to the right child node, the construction of its left sibling subtree has been completed; thus, the generation of the right child node will make full use of available information of its left sibling subtree along with its parent node.

For convenience, we define the encoder in the backbone as MathEncoder while calling the backbone as MathSolver if they have been pretrained by our proposed TCDP, which will be introduced in Section IV-B.

B. Template-Based Contrastive Distillation Pretraining

As shown in Fig. 1, although the two MWPs describe different real-world scenes, they share the same solving logic: the first number multiply by the second number. It means that if two problems have the same template, they should share the same solving logic even if they have different real-world scenes. Intuitively, the representations of them should be closer in the feature space if they share the same template, so that a solver can solve them with the same reasoning pattern. To achieve this goal, injecting solving logic to cluster the representations of the problems with the same template closer and push apart the representations of the problems with different templates is essential.

However, there are rich real-world knowledge and good linguistic representation capability in a well-trained PLM. Injecting solving logic directly will damage the linguistic representation capability of PLMs, which is also crucial for a solver to understand MWPs comprehensively. To protect the linguistic representation capability of PLMs, we also distill knowledge from well-trained PLM-based teacher encoder to retain the linguistic representation capability for comprehensive problem understanding.

Therefore, we propose TCDP to inject logic solving while retaining rich semantics, so that the representations of MWPs can be more discriminated, which can improve the solving ability of an MWP solver.

To pretrain our MathEncoder for solving logic injection and semantic retention, we first embed a PLM-based encoder into GTS [5] by replacing its GRU-based encoder and train it as a baseline. Meanwhile, we extract the fine-tuned PLM-based encoder as the teacher encoder, denoted as f^t, for linguistic semantic information retention via distilling its own knowledge into the student encoder f^s, which is initialized by the corresponding original PLM model. Here, the PLM model can be BERT [14] or Roberta [49]. Besides, we also deploy a momentum encoder f^m for contrastive template-based solving logic injection. After our proposed TCDP procedure, the student encoder f^s will be used as MathEncoder to be fine-tuned for MWP solving.

As illustrated in Fig. 2(b), for each problem P_i, our TCDP objective consists of three main components: 1) $\mathcal{L}^i_{\text{scl}}$, the template-based supervised contrastive loss for injecting solving logic by contrasting problem semantics based on student features; 2) $\mathcal{L}^i_{\text{tcl}}$, another template-based supervised contrastive loss for injecting solving logic by contrasting problem semantics anchored on teacher feature space, so that we can adjust problem semantics better according to the well-trained teacher features; and 3) $\mathcal{L}^i_{\text{sr}}$, the semantic distillation loss to distill the knowledge from f^t to f^s at token level. In these three objectives, the first two objectives aim to inject solving logic into problem representations by template-based contrastive learning, while the latter objective is adopted for linguistic semantic information retention via knowledge distillation. The final pretraining objective can be written as follows:

$$\mathcal{L}^i_{\text{tcd}} = \alpha_1 \mathcal{L}^i_{\text{scl}} + \alpha_2 \mathcal{L}^i_{\text{tcl}} + \alpha_3 \mathcal{L}^i_{\text{sr}} \tag{2}$$

where $\mathcal{L}^i_{\text{scl}}$, $\mathcal{L}^i_{\text{tcl}}$, and $\mathcal{L}^i_{\text{sr}}$ are corresponding to the template-based contrastive loss, problem-level contrastive distillation loss, and token-level knowledge distillation loss, respectively. α_1, α_2, and α_3 are three loss weights for $\mathcal{L}^i_{\text{scl}}$, $\mathcal{L}^i_{\text{tcl}}$, and $\mathcal{L}^i_{\text{sr}}$, respectively.

1) Solving Logic Injection: In MWPs, the same problem-solving logic can be described by various natural language-based short narratives. So, solving an MWP automatically requires a solver not only to understand the literal meaning of the MWP but also to mine the grounded solving logic. Therefore, how to mine the grounded solving logics and construct more discriminative solving logic-aware problem representations is important for an MWP solver to understand the MWPs and reason out the solutions. To achieve this goal, we propose a novel multiview supervised template-based contrastive pretraining method that consists of two subtasks, template-based supervised contrastive learning based on student feature space and template-based supervised contrastive learning anchored on fixed teacher feature space, as illustrated in Fig. 3. We enforce solving logic injection and construct more discriminative solving logic-aware problem representations by pulling the problem semantics with the same solving template both on student (MathEncoder) space and teacher feature space and pushing away the problem semantics with different templates.

Consider an encoded query s_i, which is outputted from the student encoder f^s, and a set of encoded samples $\{m_0, m_1, m_2, \ldots\}$ that are the keys of a dictionary Q^m, which also is a queue for storing these keys. Each key in the dictionary has its corresponding template index. Assuming that there are some positive keys m_+ in the dictionary that have the same template index with s_i and all other keys m_- are negative keys that have different template indexes with s_i, the template-based contrastive loss is a function that its value is low when s_i has the same template with its positive keys m_+ and is large among s_i and all other negative keys m_-. To generate harder positive samples and harder negative samples for better contrastive learning, we apply a dropout function with 0.2 probability as an augmentation function on the output of the momentum

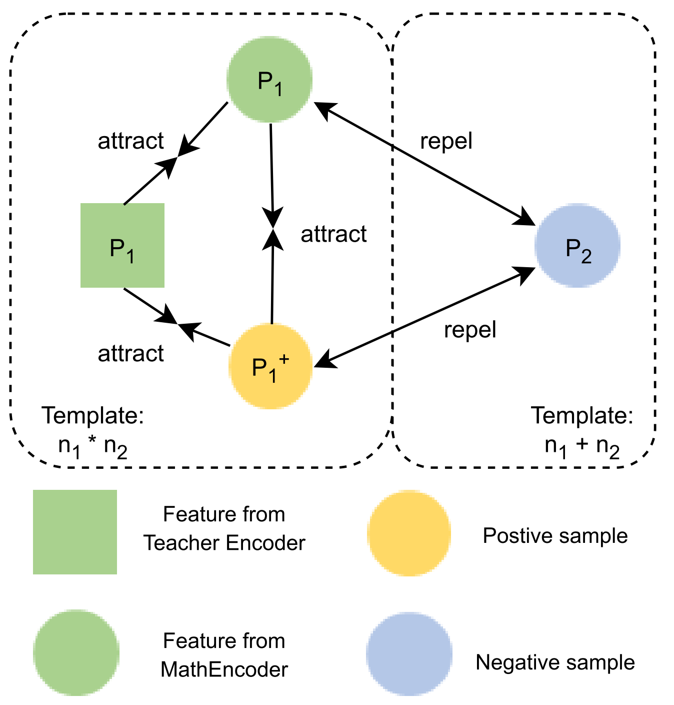


Fig. 3. Illustration of solving logic injection by contrasting problem semantics both based on student (MathEncoder) features and anchored on teacher feature space. We pull the problem semantics with the same solving template both on student (MathEncoder) space and teacher feature space and push away the problem semantics with different templates.

encoder f^m. With similarity measured by dot product, a form of our template-based contrastive loss function $\mathcal{L}^i_{\text{scl}}$, which is an extension to InfoNCE [51], can be considered as follows:

$$\mathcal{L}^i_{\text{scl}} = -\frac{1}{|m_+|}\sum_{p\in m_+}\left\{\log\frac{\exp\left(s_i\cdot m_p/\tau\right)}{\sum_{a\in Q^m}\exp(s_i\cdot m_a/\tau)}\right\} \tag{3}$$

where τ is a temperature hyperparameter.

Similarly, consider a set of encoded frozen samples $\{t_0, t_1, t_2, \ldots\}$ outputted from the teacher encoder f^t. The $\{t_0, t_1, t_2, \ldots\}$ are the keys of a dictionary Q^t for the supply of positive samples and negative samples during the template-based supervised contrastive learning anchored on fixed teacher feature space. Each key in the dictionary has its corresponding template index. We also assume that there are some positive keys t_+ in the dictionary that have the same template index with s_i, while all other keys t_- are negative keys having different template indexes with s_i. Then, $\mathcal{L}^i_{\text{tcl}}$ can be considered as follows:

$$\mathcal{L}^i_{\text{tcl}} = -\frac{1}{|t_+|}\sum_{p\in t_+}\left\{\log\frac{\exp\left(s_i\cdot t_p/\tau\right)}{\sum_{a\in Q^t}\exp(s_i\cdot t_a/\tau)}\right\} \tag{4}$$

where τ is a temperature hyperparameter.

2) Semantics Retention: However, injecting solving logic for problem understanding directly will drastically change learned real-world knowledge and linguistic semantics in the general pretraining phase with open-domain corpus. Therefore, we propose another pretrained task based on knowledge distillation for keeping the learned real-world knowledge and linguistic semantics. To distill semantic information from the teacher encoder f^t into the student encoder f^s for maintaining

Algorithm 1 Procedure of TCDP

```
Input: Teacher encoder $f^t$; Student encoder $f^s$; Momentum encoder $f^m$; Data size $N$; Training dataset $\mathcal{D}$; Problem text $P$;Expression template $T$; Weights of pretraining tasks $\alpha_1, \alpha_2, \alpha_3$; Momentum update paramater $\beta$; Contrastive queues $Q^m$ and $Q^t$;
Output: the student encoder $f^s$;
1: initialize student encoder $f^s$ with pre-trained language model
2: for all $(P_i, T_i) \in \mathcal{D}$ do
3:     $t_i = f^t(P_i)$
4:     detach $t_i$
5:     Push $t_i$ to queue $Q^t$
6: end for
7: for all $(P_i, T_i) \in \mathcal{D}$ do
8:     $s_i = f^s(P_i)$
9:     $m_i = \text{aug}(f^m(P_i))$
10:    compute $\mathcal{L}^i_{\text{scl}}$ according to Equation (3)
11:    compute $\mathcal{L}^i_{\text{tcl}}$ according to Equation (4)
12:    compute $\mathcal{L}^i_{\text{sr}}$ according to Equation (5)
13:    $\mathcal{L}^i_{\text{tcd}} = \alpha_1\mathcal{L}^i_{\text{scl}} + \alpha_2\mathcal{L}^i_{\text{tcl}} + \alpha_3\mathcal{L}^i_{\text{sr}}$
14:    update the model parameters of $f^s$ by minimizing loss $\mathcal{L}^i_{\text{tcd}}$
15:    update momentum encoder $f^m$:
       $f^m = \beta * f^m + (1-\beta) * f^s$
16:    Push $m_i$ to queue $Q^m$
17:    Pop the queue $Q^m$
18: end for
19: return the student model $\hat{f}^s$
```

fine-grained semantic information, we let the student encoder f^s mimic each token's hidden representation of the teacher encoder f^t via MSE loss

$$\mathcal{L}^i_{\text{sr}} = \frac{1}{|P_i|}\sum_{j=1}^{|P_i|}\left\|h^t_{ij} - h^s_{ij}\right\|^2 \tag{5}$$

where $|P_i|$ is the token length of a problem P_i, h^t_i is the contextual representation of the ith token of P_i from teacher encoder f^t, and h^s_i is the contextual representation of the ith token of P_i from student encoder f^s.

To summarize our TCDP procedure, we provide the algorithm pseudocode in Algorithm 1. In Algorithm 1, aug is the dropout function with 0.2 probability. We maintain a contrastive queue [34] and compute three losses: $\mathcal{L}^i_{\text{scl}}$, $\mathcal{L}^i_{\text{tcl}}$, and $\mathcal{L}^i_{\text{sr}}$. Finally, the pretrained student model $\hat{f}^s$ named MathEncoder is used for the downstream MWP task.

C. Fine-Tuning MathEncoder

Since equation label supervision is available at fine-tuning stage for MWP solving, applying MathEncoder to fine-tuning is relatively straightforward. We replace the GRU-based encoder in GTS [5] with our MathEncoder and rename it as MathSolver for distinguishing it from GTS in the experiment section. At the fine-tuning stage, our MathSolver is trained with the same hyperparameters as GTS [5] except a smaller learning rate for fine-tuning MathEncoder.

Formally, given the training dataset $\mathcal{D} = \{(P_i, S_i)\}_{i=1}^{N}$, the main learning objective for fine-tuning MathEncoder is to minimize the NLL loss function, so that it can mine the effective information and achieve good performance. Therefore, the learning objective for fine-tuning MathEncoder can be defined as follows:

$$\mathcal{L}_{\text{NLL}} = \frac{1}{N}\sum_{i=1}^{N} -\log f^{s}(S_i|P_i) \tag{6}$$

where $f^{s}(S_i|P_i)$ is the output distribution over sequences.

V. Experiments

In this section, we conduct experiments to evaluate our MathSolver with several state-of-the-art methods on two large Chinese datasets Math23K [1] and CM17K [10].

A. Datasets

In our experiments, we mainly experiment on two commonly used MWP datasets that are used in our experiments: Math23K [1] and CM17K [10]. Math23K [1] is the most frequently used large-scale dataset for MWP solving. It contains 23 162 elementary-school-level MWPs along with their well-annotated equation solution. In Math23K, the solution equation to each MWP is linear and contains only one unknown variable. CM17K [10] is another well-annotated large-scale dataset for MWP solving. It contains 17 035 MWPs with various types of expressions, including 6215 arithmetic problems, 5193 one-unknown linear problems, 3129 one-unknown nonlinear problems, and 2498 equation set problems. In all experiments, we report the results on the test sets of Math23K and CM17K.

B. Baselines

We compare our MathSolver with 12 state-of-the-art models as follows:

1) *DNS [1]:* A vanilla seq2seq model for expression generation.
2) *Math-EN [52]:* A seq2seq model with equation normalization for reducing target space.
3) *T-RNN [4]:* A recursive neural network over predicted tree-structured templates.
4) *StackDecoder [6]:* A semantically aligned MWPs solver.
5) *GROUPATT [53]:* An MWP solver borrowing the idea of multihead attention from Transformer [54].
6) *AST-Dec [55]:* An MWP solver creating an expression tree with a tree LSTM decoder.
7) *GTS [5]:* A tree-structured neural network in a goal-driven manner to generate expression trees.
8) *TSN-MD [7]:* An enhanced GTS with teacher–student distillation and multidecoder ensemble.
9) *Graph2Tree [8]:* An enhanced GTS with quantity graph.
10) *NS Solver [10]:* A neural-symbolic solver to explicitly and seamlessly incorporate different levels of symbolic constraints by auxiliary tasks.
11) *TM Generation [27]:* A template-based multitask generation model that can improve the problem-solving accuracy of mathematical word problem-solving task.
12) *Generate and Rank [26]:* A multitask framework based on a generative PLM. This model learns from its own mistakes and is able to distinguish between correct and incorrect expressions by joint training with generation and ranking.
13) *ESIB [30]:* A solver based on variational information bottleneck, which extracts essential features of the expression syntax tree while filtering latent-specific redundancy containing syntax-irrelevant features.
14) *BERT2Tree:* We deploy enhanced GTS with the BERT [14] encoder as one of our baselines whose well-trained BERT encoder is used as our teacher encoder when the PLM-based encoder in MathSolver is BERT.
15) *Roberta2Tree:* Similar to BERT2Tree, we also deploy enhanced GTS with the Roberta [49] encoder as one of our baselines whose well-trained Roberta encoder is used as our teacher encoder when the PLM-based encoder in MathSolver is Roberta.

C. Evaluation Metric

Following most of the prior works [1], [5], [8], [10], we use *answer accuracy* as the evaluation metric: if the calculated value of the predicted expression tree equals the true answer, it is thought as correct, since the predicted expression is equivalent to the target expression.

D. Implementation Details

We use PyTorch [56] to implement our model on Linux with an NVIDIA RTX2080Ti GPU card. All those words with fewer than five occurrences are converted into a special token [UNK]. In MathEncoder, the size of word embeddings and all hidden states for other layers are all set as 768, following the configuration of BERT-base [14] and Roberta-base [49]. In the decoder, the size of word embeddings and all hidden states for other layers are set as 128 and 768, respectively. In each epoch, all training data are shuffled randomly and then cut into mini-batches.

At the pretraining stage, we pretrained two MathEncoders, which are initialized separately by pretrained BERT-wwm [50] and pretrained Roberta-ext-wwm [50], so that we can validate the universality of our approach on different PLMs. The MathEncoder grounding by pretrained BERT-wwm is represented as **MathEncoder-BERT**, while the MathEncoder grounded by pretrained Roberta-ext-wwm is represented as **MathEncoder-Roberta**. Similarly, the MathSolver driven by MathEncoder-BERT is denoted as **MathSolver-BERT**, while the MathSolver driven by MathEncoder-Roberta is denoted as **MathSolver-Roberta**. All two MathEncoders are optimized by ADAM optimizor [57] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. The mini-batch size is set as 8 for both Math23K and CM17K, respectively. The initial learning rate is set as $1e^{-4}$, and we pretrain them ten epochs. The loss weights α_1, α_2, and α_3 of our TCDP loss obtained by grid search are set as 1.0, 1.0, and 1.0 for both Math23K and CM17K. The τ is set as 0.07 in both $\mathcal{L}_{\text{scl}}$ and $\mathcal{L}_{\text{tcl}}$. The momentum update parameter β is set as 0.999, and the size of contrastive queue Q equals the size of the training set.

TABLE III
ANSWER ACCURACY OF MATHSOLVER AND BASELINES ON MATH23K. NOTE THAT MATH23K DENOTES RESULTS ON THE PUBLIC TEST SET

Model	Math23K
DNS [1]	58.1
Math-EN [52]	66.7
T-RNN [4]	66.9
GROUP-ATT [53]	69.5
AST-Dec [55]	69.0
GTS [5]	75.6
TSN-MD [7]	77.4
Graph2Tree [8]	77.4
NS-Solver [10]	76.5
TM-generation [27]	85.3
Generate&Rank [26]	85.4
ESIB [30]	85.9
BERT2Tree	82.8
Roberta2Tree	84.1
MathSolver-BERT(Ours)	**84.7**
MathSolver-Roberta(Ours)	**86.2**

TABLE IV
ANSWER ACCURACY OF MATHSOLVER AND BASELINES ON CM17K

Model	CM17K
DNS [1]	15.9
GTS [5]	47.1
NS-Solver [10]	54.1
BERT2Tree	56.7
Roberta2Tree	62.6
MathSolver-BERT(Ours)	**57.1**
MathSolver-Roberta(Ours)	**63.2**

At fine-tuning stage, we use pretrained MathEncoder(*) as the initial encoder, and a randomly initialized tree-decoder is adopted as decoder for our MathSolver(*). Our MathSolver(*) is optimized by ADAM optimizer [57] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. The mini-batch size is set as 8. The initial fine-tuning learning rate is set as $1e^{-5}$ and $1e^{-4}$ for MathEncoder(*) and tree decoder in both Math23K and CM17K and then decreases to half every 25 epochs. To prevent overfitting, we set the dropout rate as 0.5 and weight decay as $1e^{-5}$. Finally, we set the beam size as 5 for expression tree generation.

E. Results on Math23K

Following prior work [8], we evaluate the performance on the test set of Math23K. As shown in Table III, we can observe that adopting a PLM-based encoder as a problem encoder can significantly improve the performance of an MWP solver on Math23K compared with those RNN-based baselines. Besides, with our TCDP for solving logic injection, our MathSolver-Roberta equipped with our MathEncoder can outperform all baselines and achieve new state-of-the-art performance on Math23K, while our MathSolver-BERT achieves competitive performance. Furthermore, comparing to the baselines with the same architecture of our MathSolvers, our MathSolver-BERT outperforms BERT2Tree over 1.9%, while our MathSolver-Roberta outperforms Roberta2Tree up to 2.0%. These experimental results on answer accuracy show the effectiveness of our pretraining approach for MWP solving.

F. Results on CM17K

To further validate the universality of our method, we also conduct experiments on another large-scale MWP dataset CM17K, which contains multiple types of MWPs and is more challenging than Math23K. The experimental results are shown in Table IV. From the experimental results, we can observe that benefiting from our TCDP, MathSolver-BERT outperforms BERT2Tree over 0.4%, while MathSolver-Roberta outperforms Roberta2Tree up to 0.6%. This shows the universality and effectiveness of our approach.

TABLE V
ABLATION OF DIFFERENT PRETRAINING TASKS ON MATH23K

$\mathcal{L}_{scl}$	$\mathcal{L}_{tcl}$	$\mathcal{L}_{sr}$	Answer Accuracy
$\surd$	$\times$	$\times$	83.1
$\times$	$\surd$	$\times$	83.5
$\times$	$\times$	$\surd$	84.9
$\surd$	$\surd$	$\times$	84.1
$\surd$	$\times$	$\surd$	84.9
$\times$	$\surd$	$\surd$	84.9
$\surd$	$\surd$	$\surd$	**86.2**

G. Ablation of Pretraining Tasks

We run an ablation study over the proposed training objectives to investigate the necessity for each of them. The MathSolver-Roberta is deployed in this ablation study. For each of these combinations, each model was trained for 100 epochs on Math23K. The experimental results are shown in Table V. We can observe that all the proposed pretraining tasks can achieve improvements individually with the help of the other two pretraining tasks. Besides, as our claim, only injecting solving logic without semantic retention will destroy the semantics entailed in the PLM-based encoder, leading to performance degradation or similar performance at the fine-tuning stage. With the help of semantic retention loss $\mathcal{L}_{\text{sr}}$ by distilling knowledge from the teacher encoder, we can outperform Robert2Tree on Math23K, especially when all three pretraining tasks are deployed simultaneously, our MathSolver-Roberta can outperform Roberta2Tree up to 2.1%. Overall, our three pretraining tasks can make the MathEncoder learn a more powerful problem representation that those encoders only used two of three pretraining tasks or without our pretraining approach, thus improving the performance of the downstream MWP solvers.

H. Analysis of Expression Tree Size

Intuitively, the larger the size of an expression tree is, the more complex the mathematical relationship of the problem is, and the more difficult it is to solve the problem. Here, we compare our fine-tuned MathSolver-Roberta and MathSolver-BERT with Roberta2Tree and BERT2Tree for investigating which kind of expression trees our pretrained MathEncoder can help MathSolver solve better. From Table VI, we can see that our MathSolver-BERT outperforms BERT2Tree on various expression tree sizes except the tree size 9. Similarly, our MathSolver-Roberta outperforms Roberta2Tree on various expression tree sizes except the tree size 9. We conjecture the data ratio of expression tree size 9 is too small, so it is hard to be improved by our TCDP. Even so, our MathSolvers still can achieve competitive performance. Although our MathSolver achieves better performance, there

TABLE VI
ANSWER ACCURACY FOR DIFFERENT EXPRESSION TREE SIZES ON MATH23K

Tree Size	BERT2Tree	MathSolver-BERT(Ours)
3-	89.1	**91.4**
5	87.0	**89.8**
7	77.0	**79.1**
9	**69.7**	60.6
11+	55.3	**59.6**
Tree Size	Roberta2Tree	MathSolver-Roberta(Ours)
3-	87.3	**92.5**
5	91.2	**91.4**
7	77.5	**80.6**
9	62.1	**62.1**
11+	51.1	**61.7**

TABLE VII
ACCURACIES OF DIFFERENT PROBABILITIES OF AUG ON MATH23K

Dropout rate	0.0	0.05	0.1	0.2	0.3
MathSolver-Roberta(Ours)	84.1	84.5	84.5	**86.2**	84.7

still is improvement room for semantic understanding and expression reasoning, especially on those problems with long expression trees.

I. Ablation of aug With Different Probabilities

To qualitatively measure how the dropout function aug works for the momentum encoder, we conduct some ablation experiments with MathSolver-Roberta on different dropout rates. The experimental results on different dropout rates are shown in Table VII. From the results, we can see that all MathSolver-Roberta models with any dropout rate of more than 0 can outperform the baseline Roberta2Tree (84.1%). This show that our aug is effective. Besides, our MathSolver-Roberta achieves the best performance on Math23K, so we choose the dropout probability 0.2 as the default value of the aug operation in our pretraining.

J. Analysis of the Accuracy on Seen Templates and Unseen Templates

To further check the generalization ability of our approach, we also analyze the answer accuracy of the seen templates and the unseen templates on Math23K test set. The seen templates mean that the templates of MWPs are overlapped with the templates in the training set, while the unseen templates mean the templates of MWPs are not in the template set of the training set. The results are shown in Table VIII. We can observe that our method can improve both the answer accuracy of the seen templates and the unseen templates when comparing with the baselines. This shows that our method is not only effective for the test samples that their templates are seen in the templates of training set, but also effective for the test samples that their templates are unseen in the templates of training set. Therefore, our method can generalize well on unseen templates due to its more discriminative problem representation ability.

Besides, although the performance of BERT-based methods is lower than the Roberta-based methods, we can observe

TABLE VIII
ANSWER ACCURACY OF MATHSOLVER AND BASELINES ON THE SEEN TEMPLATES AND THE UNSEEN TEMPLATES IN MATH23K TEST SET

Model	Seen	UnSeen
BERT2Tree	93.9	40.0
Roberta2Tree	94.8	38.3
MathSolver-BERT(Ours)	**95.2**	**44.3**
MathSolver-Roberta(Ours)	**97.4**	**43.5**

TABLE IX
ANSWER ACCURACY FOR DIFFERENT TEMPLATES ON MATH23K TEST SET

Templates	Roberta2Tree	MathSolver-Roberta(Ours)
$x = n_0 * n_1$	92.2	**96.1**
$x = n_0 / n_1$	91.2	**94.1**
$x = n_1 / n_0$	91.4	**100.0**
$x = n_0 * (1 - n_1)$	100.0	100.0
$x = n_0 * n_1 / n_2$	100.0	100.0
$x = n_0 * n_1 + n_2$	93.8	93.8
$x = n_0 * n_1 * n_2$	100.0	100.0
$x = n_1 / (1 - n_0)$	100.0	100.0
$x = (n_0 - n_2) / n_1$	94.1	94.1
$x = n_0 / (1 - n_1)$	100.0	100.0
others	80.9	**82.8**

BERT-based methods can achieve better performance than Roberta-based methods on unseen templates. We speculate this difference is due to the difference of the original pretraining ways of BERT and Roberta.

K. Analysis of the Accuracy of Different Templates

We also analyze the accuracy of different templates. We select the accuracy of the top-10 templates for comparison. The accuracy of all left templates is aggregated together and labeled as others. The results are shown in Table IX. From Table IX, we can see that our MathEncoder-Roberta can outperform Roberta2Tree on top-3 templates and others while keeping the same performance on the left templates. This shows the effectiveness of our TCDP.

L. Analysis of MathEncoder's Sentence Embedding

To qualitatively measure the effort of our in-domain pretraining, we also visualize the features of the problems of top-10 templates with t-SNE [58]. The input of the t-SNE algorithm is the mean of the hidden states at the output of the last layer of the model. As shown in Fig. 4, we can observe that with our template contrastive distillation pretraining, problem representation with different templates after our pretraining is more discriminated than those from general PLM in feature space by comparing Fig. 4(a) with (c) and comparing Fig. 4(d) with (f). Besides, we can also observe that the problem representation from our MathEncoder-Roberta and MathEncoder-BERT is more discriminated than those from Roberta2Tree and BERT2Tree, which have been fine-tuned on Math23K. These observations also show the effectiveness of our proposed template contrastive distillation pretraining.

M. Ablation of Different Encoder Architectures

Although our TCDP is proposed for the PLM-based encoder initially, to show the good generalization of our TCDP, we also

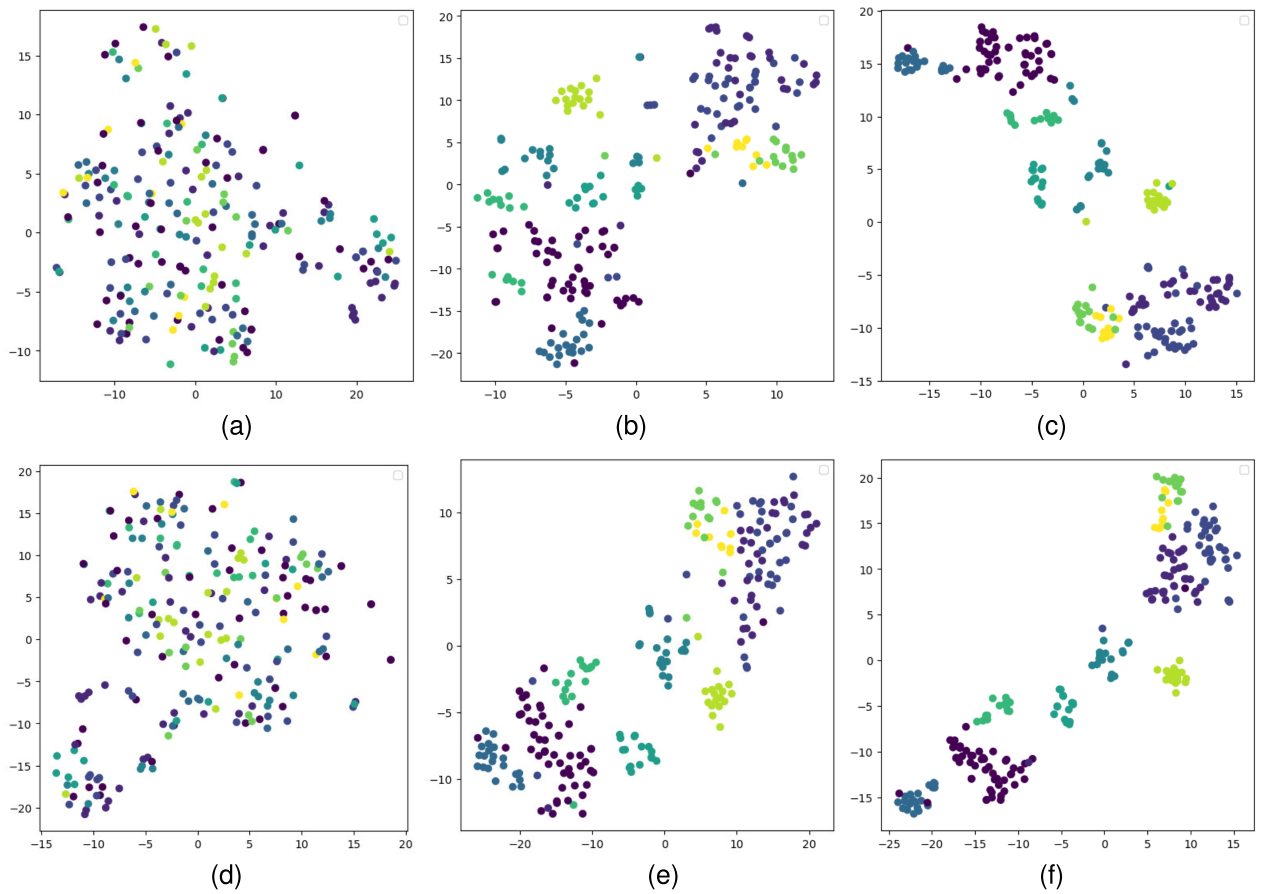


Fig. 4. Visualization on the problems of the top-10 templates in feature space before versus after template contrastive distillation pretraining (Roberta-wwm-ext versus MathEncoder-Roberta). We consider the test split in the Math23K dataset. (a) BERT-wwm-ext. (b) BERT2Tree. (c) MathEncoder-BERT (Ours). (d) Roberta-wwm-ext. (e) Roberta2Tree. (f) MathEncoder-Roberta (Ours).

TABLE X
ABLATION OF DIFFERENT PRETRAINING TASKS ON MATH23K

Encoder	*TCDP*	Answer Accuracy
GRU-based	×	75.6
	√	**76.3**
BERT-based	×	82.8
	√	**84.7**
Roberta-based	×	84.1
	√	**86.2**

conduct ablation experiments on Math23K with different encoder architectures, including GRU-based encoder [59], BERT-based encoder [14], and Roberta-based encoder [49]. The experimental results on different encoder architectures are shown in Table X. From the results in Table X, we can see that our TCDP can improve the model performance regardless of the encoder architecture. Specifically, our TCDP can improve GRU-based solver up to 0.7% while improving BERT-based solver and Roberta-based solver up to 1.9% and 2.1%, respectively. This shows that our TCDP has good generalization on different encoder architectures.

N. Case Study

Finally, we conduct a case analysis and provide three cases in Fig. 5. Benefiting from our proposed template contrastive distillation pretraining, our MathSolver-Roberta and MathSolver-BERT can generate correct equations with better problem representations, while Roberta2Tree and BERT2Tree are more prone to choose error math operations and error number words. Besides, our MathSolver-Roberta and MathSolver-BERT can be more consistent equations with ground-truth equations, while Roberta2Tree and BERT2Tree are prone to generate inconsistent equations. Overall, with the help of TCDP, our MathSolvers can solve MWP better than Roberta2Tree and BERT2Tree, which are fine-tuned on the MWP dataset directly without further in-domain pretraining.

O. Analysis of Different Data Usages

To show the strong universality of our TCDP with different data scales, we conduct some experiments with MathSolver-Roberta and Roberta2Tree on different data usages. The experimental results on different data usages are shown in Fig. 6. From the results, we can observe that our MathSolver-Roberta can outperform the corresponding baseline Roberta2Tree under various data usage settings. Especially, our MathSolver can outperform Roberta2Tree over 1.8% on answer accuracy with only 20% data are used and outperform Roberta2Tree up to 2.1% when full data are adopted. Overall, with the help of our TCDP, MathSolver can solve MWPs better under various data usage settings, showing the strong universality of our approach.

Case 1: [CLS] 有 水 果 篮 [NUM] 个， 盛 有 橘 子 的 有 [NUM] 个， 当 中 [NUM] 个 还 放 有 苹 果； [NUM] 个 水 果 篮 是 空 的，余 下 的 只 放 有 香 蕉？ 问 只 放 有 一 种 水 果 的 水 果 篮 有 多 少 个 ？[SEP] ([CLS] There are [NUM] fruit baskets , [NUM] of them contain oranges, and [NUM] of them also contain apples; [NUM] of the fruit baskets are empty, and the rest only contain bananas? How many fruit baskets are there with only one kind of fruit ? [SEP])	
BERT2Tree: $n_0 - n_2 - n_3 * n_3$ (Error)	**MathSolver-BERT (Ours):** $n_0 - n_3 - n_2$ (Correct)
Roberta2Tree: $n_0 - n_3 - n_1$ (Error)	**MathSolver-Roberta (Ours):** $n_0 - n_3 - n_2$ (Correct)
GroundTruth: $n_0 - n_3 - n_2$	
Case 2: [CLS] 李 华 买 一 顶 草 帽 用 [NUM] 元， 买 一 把 茶 壶 用 [NUM] 元， 又 买 了 一 个 热 水 瓶， [NUM] 样 东 西 正 好 用 了 [NUM] 元 ． 买 热 水 瓶 用 多 少 元？ [SEP] ([CLS] Li Hua bought a straw hat for [NUM] yuan, a teapot for [NUM] yuan, and a thermos. The [NUM] items cost exactly [NUM] yuan. How much does it cost to buy a hot water bottle? [SEP])	
BERT2Tree: $n_0 + n_1 - n_3$ (Error)	**MathSolver-BERT (Ours):** $n_3 - n_0 - n_1$ (Correct)
Roberta2Tree: $n_0 + n_1 - n_3$ (Error)	**MathSolver-Roberta (Ours):** $n_3 - n_0 - n_1$ (Correct)
GroundTruth: $n_3 - n_0 - n_1$	
Case 3: [CLS] 学 校 买 来 故 事 书 、 科 技 书 各 [NUM] 包 ． 故 事 书 每 包 [NUM] 本， 科 技 书 每 包 [NUM] 本 ． 一 共 买 来 书 多 少 本？ [SEP] ([CLS] The school bought [NUM] packs of storybooks and science and technology books. There are [NUM] books per pack and [NUM] tech books per pack. How many books did you buy in total? [SEP])	
BERT2Tree: $n_1 * n_0 + n_2 * n_0$ (Correct)	**MathSolver-BERT (Ours):** $(n_1 + n_2) * n_0$ (Correct)
Roberta2Tree: $n_1 * n_0 + n_2 * n_0$ (Correct)	**MathSolver-Roberta (Ours):** $(n_1 + n_2) * n_0$ (Correct)
GroundTruth: $(n_1 + n_2) * n_0$	

Fig. 5. Typical cases. Note that the results are represented as infix traversal of expression trees, which is more readable than prefix traversal.

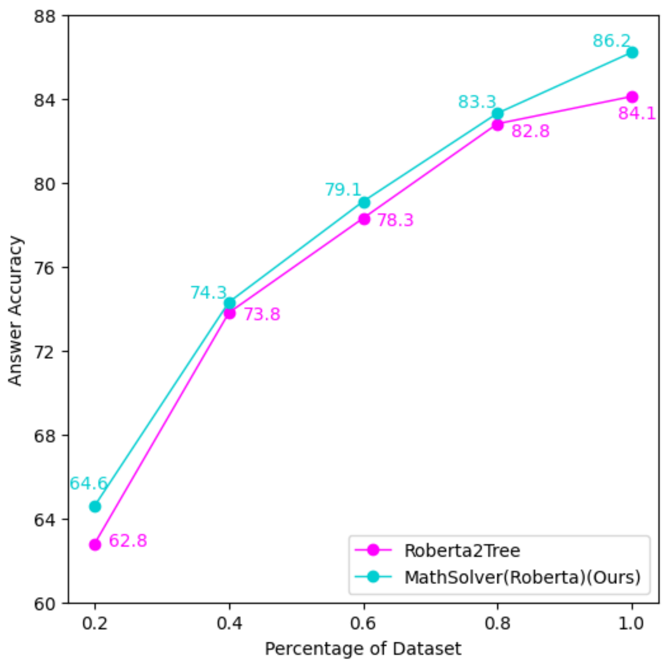


Fig. 6. Analysis of different data usages.

VI. Conclusion

In this article, we propose a TCDP approach based on a PLM-based encoder to incorporate mathematical logic knowledge by multiview contrastive learning while retaining rich real-world knowledge and high-quality semantic representation via knowledge distillation. We named the pretrained PLM-based encoder by our approach as MathEncoder. Specifically, the mathematical logic is first summarized by clustering the symbolic solution templates among MWPs and then injected into the deployed PLM-based encoder by conducting supervised contrastive learning based on the symbolic solution templates, which can represent the underlying solving logic in the problems. Meanwhile, the rich knowledge and high-quality semantic representation are retained by distilling them from a well-trained PLM-based teacher encoder into our MathEncoder. To validate the effectiveness of our pretrained MathEncoder, we construct a new solver named MathSolver by replacing the GRU-based encoder with our pretrained MathEncoder in GTS, which is a state-of-the-art MWP solver. The experimental results demonstrate that our method can carry a solver's understanding ability of MWPs to a new stage by outperforming existing state-of-the-art methods on two widely adopted benchmarks Math23K and CM17K.

Automatically solving MWPs still is a challenging AI problem, and our work opens up many thoughts for future work. First, while the results are encouraging, our work still has limitations in predicting long equations, which often match with more complex and difficult problems. For long equation prediction, we need to design a more effective pretraining method to further improve the semantic understanding ability of an MWP solver. Besides, we also need to design a more effective reasoning module to reason out long equations more efficiently. Second, our study only focuses on textual MWPs, while there are lots of MWPs, including complementary textual and graphical descriptions in real-world MWPs. In the future, we should develop new multimodal MWP datasets to extend the research boundary of MWPs and develop novel multimodal models for MWP solving.

References

[1] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2017, pp. 845–854.

[2] D. Huang, J. Liu, C.-Y. Lin, and J. Yin, "Neural math word problem solver with reinforcement learning," in *Proc. 27th Int. Conf. Comput. Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2018, pp. 213–223.

[3] L. Wang, D. Zhang, L. Gao, J. Song, L. Guo, and H. T. Shen, "MathDQN: Solving arithmetic word problems via deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5545–5552.

[4] L. Wang et al., "Template-based math word problem solvers with recursive neural networks," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 7144–7151.

[5] Z. Xie and S. Sun, "A goal-driven tree-structured neural model for math word problems," in *Proc. 28th Int. Joint Conf. Artif. Intell.* Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 5299–5305. [Online]. Available: https://www.ijcai.org/Proceedings/2019/0736.pdf, doi: 10.24963/ijcai.2019/736.

[6] T.-R. Chiang and Y.-N. Chen, "Semantically-aligned equation generation for solving and reasoning math word problems," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics. Hum. Lang. Technol.*, vol. 1. Stroudsburg, PA, USA: Association for Computational Linguistics, 2019, pp. 2656–2668.

[7] J. Zhang et al., "Teacher-student networks with multiple decoders for solving math word problem," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 4011–4017. [Online]. Available: https://www.ijcai.org/proceedings/2020/555.

[8] J. Zhang et al., "Graph-to-tree learning for solving math word problems," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 3928–3937.

[9] J. Qin, L. Lin, X. Liang, R. Zhang, and L. Lin, "Semantically-aligned universal tree-structured solver for math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 3780–3789.

[10] J. Qin, X. Liang, Y. Hong, J. Tang, and L. Lin, "Neural-symbolic solver for math word problems with auxiliary tasks," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 5870–5881.

[11] M. J. Nathan, W. Kintsch, and E. Young, "A theory of algebra-word-problem comprehension and its implications for the design of learning environments," *Cogn. Instruct.*, vol. 9, no. 4, pp. 329–389, Dec. 1992.

[12] M. Peters et al., "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, vol. 1. New Orleans, LA, USA: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://www.aclweb.org/anthology/N18-1202

[13] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, vol. 1. Minneapolis, MN, USA: Association for Computational Linguistics, Jun. 2019, p. 2.

[15] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing System*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf

[16] Z. Feng et al., "CodeBERT: A pre-trained model for programming and natural languages," in *Proc. Findings Assoc. Comput. Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, Nov. 2020, pp. 1536–1547. [Online]. Available: https://www.aclweb.org/anthology/2020.findings-emnlp.139

[17] J. Lee et al., "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020.

[18] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, "VideoBERT: A joint model for video and language representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7464–7473.

[19] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLM: Pre-training of text and layout for document image understanding," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 1192–1200.

[20] Z. Liang and X. Zhang, "Solving math word problems with teacher supervision," in *Proc. Thirtieth Int. Joint Conf. Artif. Intell. (IJCAI)*, Z.-H. Zhou, Ed. Montreal, QC, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 3522–3528. [Online]. Available: https://www.ijcai.org/proceedings/2021/0485.pdf, doi: 10.24963/ijcai.2021/485.

[21] Q. Wu, Q. Zhang, Z. Wei, and X. Huang, "Math word problem solving with explicit numerical values," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2021, pp. 5859–5869.

[22] Y. Cao, F. Hong, H. Li, and P. Luo, "A bottom-up DAG structure extraction model for math word problems," in *Proc. 35th AAAI Conf. Artif.*, 2021, pp. 39–46.

[23] X. Lin et al., "HMS: A hierarchical solver with dependency-enhanced understanding for math word problem," in *Proc. 35th AAAI Conf. Artif.*, 2021, pp. 4232–4240.

[24] Q. Wu, Q. Zhang, and Z. Wei, "An edge-enhanced hierarchical graph-to-tree network for math word problem solving," in *Proc. Findings Assoc. Comput. Linguistics (EMNLP)*, 2021, pp. 1473–1482.

[25] S. Huang, J. Wang, J. Xu, D. Cao, and M. Yang, "Recall and learn: A memory-augmented solver for math word problems," in *Proc. Findings Assoc. Comput. Linguistics (EMNLP)*, 2021, pp. 786–796.

[26] J. Shen et al., "Generate & rank: A multi-task framework for math word problems," in *Proc. Findings Assoc. Comput. Linguistics (EMNLP)*, 2021, pp. 2269–2279.

[27] D. Lee, K. S. Ki, B. Kim, and G. Gweon, "TM-generation model: A template-based method for automatically solving mathematical word problems," *J. Supercomput.*, vol. 77, no. 12, pp. 14583–14599, Dec. 2021.

[28] Y. Hong, Q. Li, D. Ciao, S. Huang, and S.-C. Zhu, "Learning by fixing: Solving math word problems with weak supervision," 2020, *arXiv:2012.10582*.

[29] Y. Hong, Q. Li, R. Gong, D. Ciao, S. Huang, and S.-C. Zhu, "SMART: A situation model for algebra story problems via attributed grammar," 2020, *arXiv:2012.14011*.

[30] J. Xiong, C. Li, M. Yang, X. Hu, and B. Hu, "Expression syntax information bottleneck for math word problems," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 2166–2171.

[31] J. Yang et al., "Towards making the most of BERT in neural machine translation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 9378–9385.

[32] J. Zhu et al., "Incorporating BERT into neural machine translation," 2020, *arXiv:2002.06823*.

[33] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2006, pp. 1735–1742.

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9729–9738.

[35] Z. Yang, Y. Cheng, Y. Liu, and M. Sun, "Reducing word omission errors in neural machine translation: A contrastive learning approach," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 6191–6196.

[36] K. Clark, M. T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," 2020, *arXiv:2003.10555*.

[37] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6894–6910. [Online]. Available: https://aclanthology.org/2021.emnlp-main.552

[38] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," 2019, *arXiv:1910.10699*.

[39] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[40] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4794–4802.

[41] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, "Structured knowledge distillation for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2604–2613.

[42] S. Liang, Z. Huang, M. Liang, and H. Yang, "Instance enhancement batch normalization: An adaptive regulator of batch noise," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4819–4827.
[43] Z. Huang, S. Liang, M. Liang, and H. Yang, "DIANet: Dense-and-implicit attention network," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4206–4214.
[44] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Austin, TX, USA: Association for Computational Linguistics, Nov. 2016, pp. 1317–1327. [Online]. Available: https://aclanthology.org/D16-1139
[45] M. Hu et al., "Attention-guided answer distillation for machine reading comprehension," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Brussels, Belgium: Association for Computational Linguistics, Oct./Nov. 2018, pp. 2077–2086. [Online]. Available: https://aclanthology.org/D18-1232
[46] K. Yue, J. Deng, and F. Zhou, "Matching guided distillation," in *Proc. Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, 2020, pp. 312–328. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-58555-6_19, doi: 10.1007/978-3-030-58555-6_19.
[47] D. Chen et al., "Cross-layer distillation with semantic calibration," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 8, pp. 7028–7036.
[48] L. Liu et al., "Exploring inter-channel correlation for diversity-preserved knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 8271–8280.
[49] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
[50] Y. Cui, W. Che, T. Liu, B. Qin, S. Wang, and G. Hu, "Revisiting pre-trained models for Chinese natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Findings*. Stroudsburg, PA, USA: Association for Computational Linguistics, Nov. 2020, pp. 657–668. [Online]. Available: https://www.aclweb.org/anthology/2020.findings-emnlp.58
[51] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.
[52] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a math word problem to a expression tree," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2018, pp. 1064–1069.
[53] J. Li, L. Wang, J. Zhang, Y. Wang, B. T. Dai, and D. Zhang, "Modeling intra-relation in math word problems with different functional multi-head attentions," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 6162–6167.
[54] A. Vaswani et al., "Attention is all you need," 2017, *arXiv:1706.03762*.
[55] Q. Liu, W. Guan, S. Li, and D. Kawahara, "Tree-structured decoding for solving math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 2370–2379.
[56] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[58] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.
[59] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1724–1734.

Jinghui Qin received the B.S. and M.A.Eng. degrees from the School of Software, Sun Yat-sen University, Guangzhou, China, in 2012 and 2014, respectively, and the Ph.D. degree from the School of Data and Computer Science, Sun Yat-sen University, in 2020.

He is currently a Post-Doctoral Fellow with Sun Yat-sen University. His research interests include nature language processing, machine learning, and computer vision.

Dr. Qin has been serving as a Reviewer for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON BROADCASTING, *Neurocomputing*, *Computer Vision and Image Understanding*, and *Bioinformatics*.

Zhicheng Yang received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2020, where he is currently pursuing the M.S. degree.

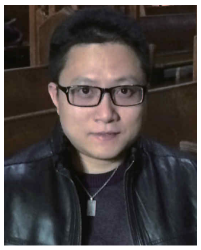
His research interests include natural language processing.

Jiaqi Chen received the B.S. degree in electronic information engineering from Xidian University, Xi'an, China, in 2019, and the M.S. degree in computer technology from Sun Yat-sen University, Guangzhou, China, in 2021.

His research interests include natural language processing and computer vision.

Xiaodan Liang received the Ph.D. degree from Sun Yat-sen University, Guangzhou, China, in 2016, advised by Liang Lin.

She was a Post-Doctoral Researcher with the Machine Learning Department, Carnegie Mellon University, working with Prof. Eric Xing, from 2016 to 2018. She is currently an Associate Professor with Sun Yat-sen University, Guangzhou, China. She has published several cutting-edge projects on human-related analysis, including human parsing, pedestrian detection and instance segmentation, 2-D/3-D human pose estimation, and activity recognition.



Liang Lin (Senior Member, IEEE) is the Excellent Young Scientist of the National Natural Science Foundation of China. From 2008 to 2010, he was a Post-Doctoral Fellow with the University of California at Los Angeles, Los Angeles, CA, USA. From 2014 to 2015, as a Senior Visiting Scholar, he was with Hong Kong Polytechnic University, Hong Kong, and the Chinese University of Hong Kong, Hong Kong. He is currently a Full Professor with Sun Yat-sen University. He has authored and coauthored more than 100 papers in top-tier academic journals and conferences.

Mr. Lin is a fellow of IET. He served as an Area/Session Chair for numerous conferences, including ICME, ACCV, and ICMR. He was the recipient of the Best Paper Runners-Up Award at ACM NPAR 2010, a Google Faculty Award in 2012, the Best Paper Diamond Award at IEEE ICME 2017, and the Hong Kong Scholars Award in 2014. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS, *The Visual Computer*, and *Neurocomputing*.