Teacher-Student Networks with Multiple Decoders for Solving Math Word Problem

Jipeng Zhang[1,2], **Roy Ka-Wei Lee**[3], **Ee-Peng Lim**[2], **Wei Qin**[2,4]
Lei Wang[2], **Jie Shao**[1,5,*] and **Qianru Sun**[2,*]
[1]Center for Future Media, University of Electronic Science and Technology of China
[2]School of Information Systems, Singapore Management University
[3]Department of Computer Science, University of Saskatchewan
[4]School of Computer Science and Information Engineering, Hefei University of Technology
[5]Sichuan Artificial Intelligence Research Institute
zhangjipeng20@std.uestc.edu.cn, eplim@smu.edu.sg, roylee@cs.usask.ca qinwei.hfut@gmail.com,
lei.wang.2019@phdcs.smu.edu.sg, shaojie@uestc.edu.cn, qianrusun@smu.edu.sg

Abstract

Math word problem (MWP) is challenging due to the limitation of training data where only one "standard" solution is provided. MWP models often fit the solution rather than truly understand or solve the problem. The generalization of models (to diverse word scenarios) is thus limited. To address this problem, we propose a novel approach we call TSN-MD that leverages a teacher network to integrate the knowledge of equivalent solution expressions such as to better regularize the learning behavior of the student network. In addition, we introduce the multiple-decoder student network to generate multiple candidate solution expressions by which the final answer is voted. In experiments, we conduct extensive comparisons and ablative studies on two large-scale MWP benchmarks, and show that using TSN-MD can surpass the state-of-the-art works by large margins. Intriguingly, the visualization results demonstrate that TSN-MD not only produces correct answers but also generates diverse equivalent expressions for the solution[1].

1 Introduction

Introducing the math word problem (MWP) aims to develop machine learning models, i.e., automated solvers, to answer mathematical questions given in texts. It has been an interesting natural language understanding task for more than a half decade [Bobrow, 1964] and attracted much attention in recent years [Koncel-Kedziorski *et al.*, 2015; Shi *et al.*, 2015; Huang *et al.*, 2017; Wang *et al.*, 2018b; Liu *et al.*, 2019; Xie and Sun, 2019]. The MWP solving task is challenging due to three reasons. (1) Math problems are often stated in diverse word scenarios. (2) There are strong semantic gaps between language logic and math expression. (3) It is not obvious how to model the mathematical laws, e.g., commutative law of summation and multiplication, in machine learning paradigms.

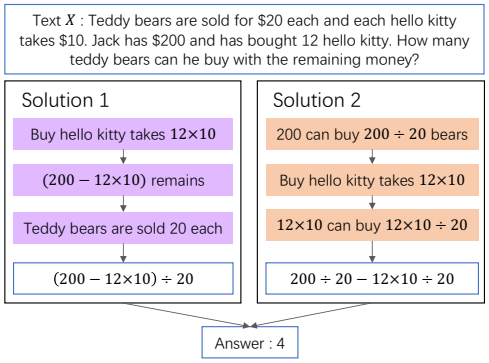

Figure 1: Multiple solution expressions for an MWP and intermediate steps for problem solving.

Typically, an MWP is composed of a problem description text, a solution expression in math and a final answer as a number. An example is given in Figure 1. With such training examples, MWP solvers should have two objectives to optimize, i.e., the generation loss of answers, as well as the loss of solution expressions which indirectly minimizes the answer loss. While, most existing approaches use only the latter objective. In specific, their solvers are required to identify the relevant numbers and compose solution expression from the inferred relations between/among those numbers [Wang *et al.*, 2017; Wang *et al.*, 2018a]. The state-of-the-art framework is based on the deep generative neural network composed by an encoder and a decoder [Xie and Sun, 2019]. The encoder encodes the problem text into latent embedding which is then fed into the decoder to generate the solution expression. The solution expression is represented by either a sequential [Wang *et al.*, 2017; Wang *et al.*, 2018a; Chiang and Chen, 2019] or a tree structure [Liu *et al.*, 2019; Xie and Sun, 2019]. Most of such frameworks deploy single decoder and fit to the only one solution expression given on

*Corresponding Author
[1]Code and datasets: https://github.com/2003pro/TSN-MD

datasets. It is, therefore, hard for them to generate diverse but equivalent solution expressions, limiting the generalization to various word scenarios.

In order to overcome these limitations, we propose a novel and effective MWP solver which we call Teacher-Student Networks with Multiple Decoders (TSN-MD). We pre-train the teacher network to guide the learning behavior of the student network with the same training data. The intuition is that the teacher observed through all solutions including those with equivalent solution expressions under mathematical laws. It gains the knowledge of regularizing the learning behaviours of the student network. More intuitively, when student predicts an equivalent solution expression to the ground-truth but causes an undesirable loss, the teacher network will help to amend this mistake. In addition, to encourage high-diversity predictions (i.e., solution expressions) from the student network, we use multiple decoders, e.g., one decoder is conventional and the other one has its input embedding perturbed by a Gaussian noise.

Contributions. Our main contributions are three-fold. (i) A novel and effective teacher-student approach that addresses the key limitations of solution expressions existing on the MWP data. (ii) A multiple-decoder architecture that works for our teacher-student networks to improve the diversity of generated expressions. (iii) Extensive experiments and ablative studies on two large-scale MWP benchmarks – Math23k [Wang *et al.*, 2017] and MAWPS [Koncel-Kedziorski *et al.*, 2016], showing the superiority of the proposed approach over related works. In addition, the visualized results show that our model is really generating diverse and equivalent expressions of the solution, explicitly revealing the reason behind our superiority.

2 Related Work

Math Word Problems. In the context of math word problems (MWP), the algorithms are designed to calculate the results of mathematical questions given in a natural language. Therefore, the deep learning community tends to regard MWP as one sort of natural language process (NLP) task. Most of these existing methods adopt an encoder-decoder framework, where the encoder is given the representations of problem text and the decoder generates the corresponding solution expression. For instance, Wang *et al.* [2017] proposed a large-scale MWP dataset and applied a vanilla sequence-to-sequence (Seq2Seq) model to translate the problem text to a solution expression.

The later works took more exclusive characteristics of the problem into account progressively. For instance, considering that the composition of the equations is based on some clear regulars, Wang *et al.* [2018a] introduced an equation normalization method to normalize the duplicated equations. Inspired by the stack mechanism in the data structure, Chiang and Chen [2019] introduced the copy mechanism with a stack. Li *et al.* [2019] introduced MWP specific priors into the intermediate representations.

The recent studies have attempted to improve the generation of solution expressions using sequence-to-tree (Seq2Tree) models [Wang *et al.*, 2018a; Chiang and Chen, 2019; Wang *et al.*, 2019; Liu *et al.*, 2019; Xie and Sun, 2019]. For example, the Goal-driven Tree Structure model (GTS) [Xie and Sun, 2019] achieved the state-of-the-art performance by guiding the model to generate specific solution equations using a tree structure.

However, all these methods suffer from same limitation. In the beginning, researchers [Wang *et al.*, 2018a] suggest that the equation search space in the decoding process is large with much ambiguity in solution expression. Therefore, these models are optimized to generate a fixed target solution expression instead of finding the correct answers. This constraint to fit an MWP to single solution expression ignores the possibility that an MWP can be solved by multiple solution expressions. In many cases, the model will falsely penalize the correct generated solution expression, which introduces undesired bias.

Knowledge Distillation. Knowledge distillation (KD) was first introduced by Hinton *et al.* [2015]. Aiming at compressing neural networks, it transfers knowledge from a complicated high-performance teacher model to a simple student model without losing too much performance. To this end, Hinton *et al.* [2015] reshaped the training procedure, where they regarded the predictions of the teacher model as "soft labels" and trained the student model to fit the soft labels. Romero *et al.* [2015] trained the student model to mimic not only the final outputs of the teacher model but also the intermediate feature map of the teacher model. This can help to narrow the performance gap between teacher and student models.

While knowledge distillation was firstly proposed for model compression, the intuition behind the soft labels was applied widely in different tasks and domains. Furlanello *et al.* [2018] and Zhang *et al.* [2018] found that applying the soft labels as the training target can help the student model achieve better performance. Zagoruyko and Komodakis [2017] showed that the teacher-student learning framework managed to distill the attention map from a strong teacher model to the student model.

Some works also explored applying knowledge distillation in natural language processing tasks. Kim and Rush [2016] introduced a sequence-level distillation framework to distill the structure loss for neural machine translation. Knowledge distillation was also applied in other tasks such as text classification [Zagoruyko and Komodakis, 2017], parsing [Kuncoro *et al.*, 2016] and machine reading comprehension [Hu *et al.*, 2018]. To the best of our knowledge, this is the first work that explores the usage of knowledge distillation on MWP task.

Multiple-Decoder Networks. Multiple-decoder networks are mainly applied in machine translation and image captioning tasks to generate diverse outputs [Jacobs *et al.*, 1991; Eigen *et al.*, 2014]. For instance, Yang *et al.* [2018] proposed to use multiple softmax heads after the encoding by recurrent neural networks. Shen *et al.* [2019] and He *et al.* [2018] used multiple decoders with uniform mixing coefficient to improve diversity in machine translation. In this paper, we adapt multiple decoders to generate diverse solution expressions for a given MWP.

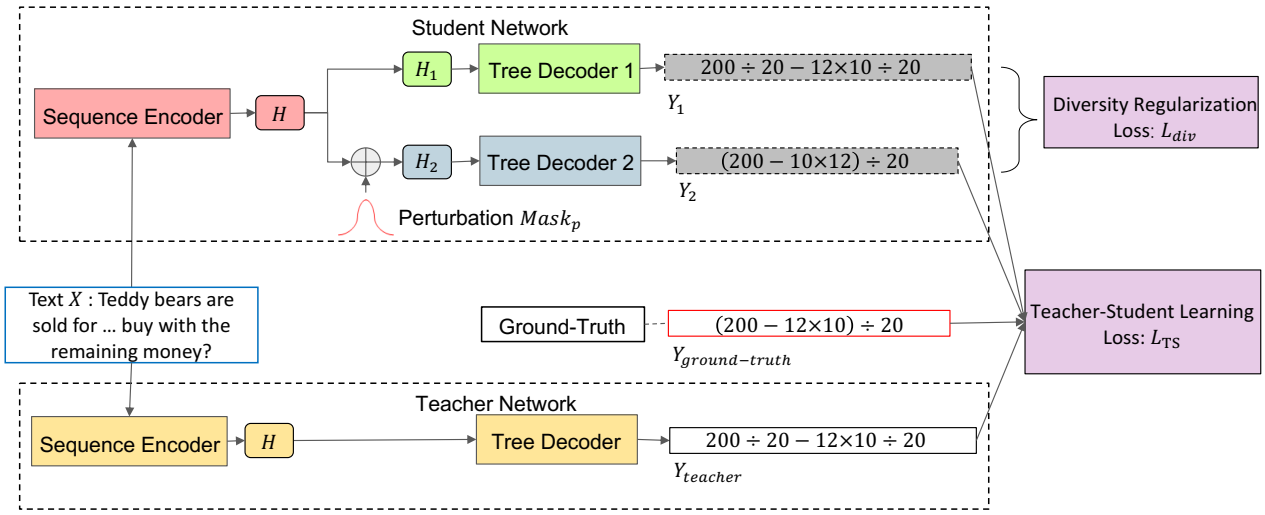

Figure 2: Overview of the proposed approach. The teacher network is trained in prior with baseline architectures — GTS. After that, the student network is fed with the problem text X to generate intermediate embedding H. H_1 is copied from H and H_2 is computed by perturbing H. Then, they are input to Tree Decoder 1 and Tree Decoder 2, respectively, which output solution expressions Y_1 and Y_2. In the meantime, the teacher network outputs a soft prediction $Y_{teacher}$ for the same instance. Finally, ground-truth is used to calculate losses.

3 Teacher-Student Networks with Multiple Decoders (TSN-MD)

As illustrated in Figure 2, the proposed TSN-MD is composed of a teacher network and a student network. The student network includes an encoder and two decoders. Teacher network is pre-trained (in the baseline way) and it regularizes the learning behaviour of the student network by reducing the loss when student predicts solution expressions that are mathematically equivalent to the ground-truth. Here, we note that student network has one more decoder than teacher network in order to output more diverse solution expressions.

Notations. We denote a math word problem X by a set of n words $\{x_i\}_{i=1}^n$, whose word embedding vectors $\{e_i\}_{i=1}^n$ are obtained through a linear projection of word embedding matrix E, i.e., $e_i = E\, x_i^X$, and its corresponding ground-truth solution expression by Y.

Considering choosing important numbers from problem text, we follow Wang *et al.* [2017] which proposed to mapping numbers into special tokens following these two rules: 1) All numbers appeared in the problem text are determined if they are significant numbers, which means they will be used in the solution expression. This is the so-called Significant Number Identify (SNI) mechanism [Wang *et al.*, 2017]; 2) All the recognized significant numbers in a problem X are mapped to a list of mapped position symbols $\{n_1, ..., n_l\}$ according to their appearing orders in the problem text. Simultaneously, numbers in solution expression are then mapped as the position symbols $\{n_1, ..., n_l\}$. For example, the position symbols and mapped solution expression of the problem in Figure 1 are $\{n_1 = 20, n_2 = 10, n_3 = 200, n_4 = 12\}$ and $(n_3 - n_4 \times n_2) \div n_1$ respectively. Then, solution expressions are normalized to prefix expression following the pre-defined rules.

Our baseline model is the goal driven tree structure solver (GTS) [Xie and Sun, 2019]. Based on its architecture, we introduce how to build our TSN-MD — the teacher-student networks with one conventional decoder as well as one perturbed decoder.

3.1 Teacher-Student Networks

It is often the case that there exists correct solution expressions different from the annotation solution expression. Previous models will falsely penalize these correct solutions as the training objective only fits the fixed solution expression.

For example, the label solution expression of the problem in Figure 1 is $(200 - 12 \times 10) \div 20$, although $200 \div 20 - 12 \times 10 \div 20$ is also a correct solution expression and can compute the right answer. The previous training objective will undoubtedly penalize this kind of generation results.

In the empirical analysis on GTS's generated solution expression, we found that the number of exactly matched generated solution expressions is lower than generated solution expressions that can get the correct answer. That is to say, GTS model can find some correct solution expressions different from annotated ones. This also demonstrates the existence of the undesired bias in training objective.

A simplest solution is to improve solution annotations and add extra training examples. However, the current datasets all only give the single solution expression annotation. Here, based on the interesting fact that GTS model can find some correct new solution expressions, we propose to use teacher-student learning, which is also known as knowledge distillation, to transfer these kinds of multi-solution knowledge to the student model.

More concretely, consider the classical MWP solving setting where we have training set consisting of tuples of problem texts and solution expressions (x, y) with possible classes

V. We aim at obtaining the function $f(x) : X \mapsto Y$, which can achieve good results in unseen data. Commonly, the function $f(x)$ is parametrized as a neural network $f(x, \theta)$. θ represents parameter of this neural network. The common training criteria is to minimize negative log-likelihood (NLL) loss for each example from the training data:

$$L_{NLL}(\theta) = -\sum_{k=1}^{|V|} 1\{y = k\} log\ p(y = k|x; \theta)$$

where 1 is the indicator function and p is the output distribution from the model with parameter θ. From the viewpoint of cross entropy, this loss function can be regarded as the cross entropy between the degenerate data distribution (with all the probabilities focusing on one class) and the model output distribution.

In teacher-student network, we assume to have access to a well-trained teacher network parametrized with θ_T. More precisely, we choose GTS [Xie and Sun, 2019] as our teacher network. The output distribution $q(y|x; \theta_T)$ of teacher network can provide extra source of training information and this has been proved in previous work. Thus, we tried to minimize the cross entropy between our network's output distribution and teacher network's output distribution:

$$L_{KD}(\theta, \theta_T) = -\sum_{k=1}^{|V|} q(y = k|x; \theta_T) log\ p(y = k|x; \theta)$$

The student network should not only aim at fitting the teacher network's output, which has the risk of being limited by the teacher's performance, but also learn from the original label. Thus, we interpolate between the two loss functions:

$$L_{TS}(\theta; \theta_T) = (1 - \alpha)L_{NLL}(\theta) + \alpha L_{KD}(\theta, \theta_T)$$

where α is the interpolation coefficient.

3.2 Multiple Decoders

If only simply borrow the idea of teacher-student learning, we may suffer from false guidance given by teacher model. After all, teacher model still predicts false results in test set. In other words, if the teacher model falls into some local optimization point, the student model possibly also falls into the similar local optimization points. In addition, we suggest teacher-student learning is not enough to overcome the issue. In order to alleviate this extra undesired bias given by teacher model, we propose to use multiple decoder structure to further increase the possibility of getting diverse solutions.

More precisely, this novel multiple decoder structure is composed with one conventional decoder and many perturbed decoders. This can improve both prediction accuracy and diversity.

The Overall Multiple Decoder Architecture

Our TSN-MD model naturally incorporates diversity in the architecture both during training and inference. Furthermore, we try to encourage the diversity by feeding these decoders with different latent variable inputs.

Perturbations to Get Multiple Decoders. In the student network, all of the decoders share the encoder, i.e., this model requires processing the input sentence only once. Aiming at getting different outputs from different decoders, we try to feed latent variables to different decoders. Here, except for the conventional decoder which is directly fed with encoder output, we introduce perturbed decoder. In this perturbed decoder, perturbation masks $Mask_p$ are added to the output of encoder H to get different input H_l for perturbed decoder:

$$H_l = Mask_p \otimes H$$

where $\otimes$ represents element-wise production. In order to get $Mask_p$, we first define the mask rate P_{mask}. Next, we sample P_{mask} percent region in H according to Gaussian distribution. we define a zero matrix $Mask_{zero}$ with the same shape of H and assign one to positions that locate in the region.

Diversity Regularization. After getting the masked latent variable set $H = \{h_1, h_2, \cdots\}$, we feed h_l to the l-th tree decoder and get outputs y_l. In order to encourage different decoders to generate different results, we introduce the diverse regularization item. More specifically, we adopt the cosine similarity to quantify the difference of different outputs generated by different decoders. Our intuition is consistent with promoting the diversity of generation. For two output solution expressions (y_l, y_{l+1}) , if they are far apart, the regularization item should give larger penalty:

$$L_{div,t} = 1 + s_{cos}(y_{l,t}, y_{l_1,t})$$

where t represents the position of word, and l and l_1 mean two outputs from two different decoders. The regularization loss on the completed sequence is:

$$L_{div} = \sum_{l,l_1} \sum_{t}^{T} L_{div,t}$$

where T means the length of generated sequence.

Separate Beam Search per Component. When it comes the inference, separate beam search is executed per mixture component. As different components tend to focus on different information and grab different patterns, performing independent beam search encourages the diversity of generation. Note that we have $|H|$ different components and the beam size as b. Overall, this requires processing $|H| \times b$ candidates at each step.

It is worth noticing that the solver can only output one solution expression. Therefore, we vote to get the output solution expression according to the prediction score given by different decoders.

3.3 The Overall Training Objectives

In the stage of training teacher network, we follow GTS training process [Xie and Sun, 2019] and save the output classification probability distribution of training set as the soft label. In the stage of training student network, we feed both soft label and one-hot hard label to the model. Also, the diversity regularization term and teacher-student learning objectives are combined. Consequently, the overall objective

function is formed as:

$$L = \beta L_{div} + \sum_{i}^{N} L_{TS,N}$$

where β is the weight of the regularization term and N is the number of decoders.

4 Experiments

In this section, we conduct extensive ablation studies to validate the effectiveness of our contributing components — teacher-student networks and multiple decoders. We also demonstrate a case study to show the effects by using our proposed methods.

Datasets. Two large-scale MWP benchmarks, i.e., Math23K [Wang *et al.*, 2017] and MAWPS [Koncel-Kedziorski *et al.*, 2016], are used in our experiments.

Baselines. We compare TSN-MD to following baseline and state-of-the-art models: **DNS** [Wang *et al.*, 2017] uses a vanilla seq2seq model to generate expressions. **Math-EN** [Wang *et al.*, 2018a] proposes an equation normalization to reduce target space. **T-RNN** [Wang *et al.*, 2019] applies recursive neural networks over the predicted tree-structure templates. **S-Aligned** [Chiang and Chen, 2019] introduces a stack structure decoder to track the semantic meanings of operands. **GROUP-ATT** [Li *et al.*, 2019] applies the idea of multi-head attentions from Transformer [Vaswani *et al.*, 2017]. **AST-Dec** [Liu *et al.*, 2019] proposes to generate expression tree by a tree LSTM decoder. **GTS** [Xie and Sun, 2019] develops a tree structured neural network in a goal-driven manner.

Implementation Details and Evaluation Metric In TSN-MD, we use a word embedding with 128 units. The dimension of the hidden state for all the other layers are set to 512. Besides, we use two heads decoder in TSN-MD. Our model is trained for 80 epochs. Mini-batch size and dropout rate are set to 64 and 0.45, respectively. For optimizer, we use Adam with learning rate set to 0.001, $\beta_1 = 0.94$ and $\beta_2 = 0.99$, and the learning rate will be halved every 20 epochs. Also, We use a beam size of 5 in beam search.

4.1 Overall Results

	MAWPS	Math23K	Math23K*
DNS	59.5	-	58.1
Math-EN	69.2	66.7	-
T-RNN	66.8	66.9	-
S-Aligned	-	-	65.8
GROUP-ATT	76.1	69.5	66.9
AST-Dec	-	69.0	-
GTS	82.6	75.6	74.3
TSN-MD	**84.4**	**77.4**	**75.1**

Table 1: Solution accuracy of TSN-MD and various baselines. Note that Math23K denote results on public test set and Math23K* denote 5-fold cross-validation. For the MAWPS dataset, the models are evaluated with 5-fold cross-validation.

The main results are shown in Table 1. For evaluation with test set on Math23k, the single model performance achieved by our proposed TSN-MD achieves the state-of-the-art. As the code for GTS is made available[2], we implemented GTS and tested it on all dataset settings. The empirical results show the effectiveness of teacher-student learning and mixture of decoder structure. On all the benchmark datasets, our model achieves the new state-of-the-art performance.

4.2 Solution Expression Prediction Analysis

We report the accuracy of solution expression prediction by the sequence-to-sequence model (Math-EN), sequence-to-tree model (GTS) and our TSN-MD in Table 2. If the predicted solution expression exactly matches the annotated solution, we consider it a positive example. Here, we can compute the number of alternative solution that each model can provide, which can partly tell the degree of diversity of a model. It can be observed that the accuracy of solution expression generation is lower than the final accuracy of problem solving. This reflects that these MWP solvers have spontaneously learned to adapt the situation that a certain problem may have multiple solutions.

	Equation-ac	Answer-ac
Math-EN	60.1	66.7
GTS	64.8	75.6
TSN-MD	65.8	77.4

Table 2: Accuracy of equation generation.

4.3 Ablation Study and Parameter Analysis

Effect of Different Components

To better understand the performance contributed by different components adopted in this paper, we perform a series of ablation tests. All the performance in ablation study is evaluated by the test set on Math23k.

	Math23K
TSN-MD	77.4
GTS	75.6
GTS + Teacher-Student Learning + Multi-Decoder	76.8
GTS + Teacher-Student Learning	76.8
GTS + Multi-Decoder + regularization	76.2
GTS + Multi-Decoder	75.9

Table 3: Solution accuracy with various model configurations in TSN-MD. Here BASE means GTS in Table 1.

We investigate the effects from different modules. The results of different MWP solver variants on Math23K are shown in Table 3. Taking GTS as the base model, we try to verify the effectiveness of proposed techniques. First, we evaluate the vanilla teacher-student network by removing multiple decoder structure. The results show that the teacher-student network contributes a lot to improve the performance. Next, we observe that it is beneficial to use multiple decoders. Using 2 decoders with diversity regularization item, the accuracy improved by approximately 0.6%. We

[2]https://github.com/ShichaoSun/math_Seq2Tree

also notice that it is necessary to include the diversity regularization techniques. We ablate the regulation item both for the full model and the model with only multiple decoders structure. The results in Table 3 show performance degradation to different extents. In addition, we have verified that "GTS+Teacher-Student Learning+Multi-Decoder" and "GTS+Teacher-Student Learning" in Table 3 solve different parts of MWPs while sharing accuracy. It is not fair to infer that "Multi-Decoder" is unnecessary from this comparison. Compared with single decoder, the model can get more different solutions with multiple decoders. In this case, diversity regularization item can have larger influence. Therefore, although the impact on the multiple decoders without teacher-student network is relatively small, the influence on the full model is large.

Parameter Analysis

	Dec_1	Dec_2	Dec_3	Comb
TSN-MD static mask				
1% mask	75.8	76.4	-	76.3
3% mask	76.2	74.6	-	75.8
5% mask	77	76.7	-	77.4
10% mask	74.7	73.8	-	75.1
TSN-MD dynamic mask				
1% mask	75.1	74.2	-	75.2
3% mask	74.6	75.3	-	75.2
5% mask	75.6	75.8	-	76
10% mask	74.7	75.1	-	75.6
TSN-MD static 3 decoder	76.2	74.9	74.9	76.3
TSN-MD dynamic 3 decoder	75.3	75.1	75.1	75.4

Table 4: Accuracy for various hyperparameter configurations.

Mask Rate. The percentage of mask rate is a tune-able hyperparameter in our TSN-MD. Thus, we investigate the effect of the percentage of mask rate on our model's performance. Table 4 suggests 5% achieves the best performance.

Dynamic or Static. We explore whether to fix the mask in the beginning (static) or dynamically change it during training (dynamic). The results show that in most cases, the static strategy can get better results. We argue that as the dynamic mask keeps changing the decoder inputs, it is harder to get effective information.

Number of Decoders. At last, we investigate the effect of increasing number of decoders. As adding decoders will bring a lot of extra computation cost, we only examine the effect of adding to 3 decoders. The experiment results show that the 2 decoder is the best configuration.

Besides, Table 4 has also shown the importance of random perturbation.

4.4 Case Study

We present several generation examples sampled from the Math23k dataset in Table 5. In the first example, the problem is asking the proportion of the fired ones. However, GTS falsely predicts the solution expression to compute the proportion of the remaining members and Decoder1 gets the same error. Our model gives the right solution with the help of Decoder2 exploring one more possibility. For the second example, GTS generates a new solution different from the ground-truth. Two decoders also give diverse correct solutions. For the third example, there are key words in the problem text like "quotient" and "remainder" while the solution actually needs "$\times$" and "$+$". This case asks for model to handle the confusing descriptions. Our TSN-MD can better deal with this situation.

Problem Text:	There used to be 120 staff members in a government department. Currently, the number of staff members is 90. What is percentage of fired people over original total number of staff members?
GT Solution:	$(120 - 90) \div 120$
Teacher (GTS):	$90 \div 120$
Decoder1:	$90 \div 120$
Decoder2:	$(120 - 90) \div 120$
TSN-MD:	$(120 - 90) \div 120$
Problem Text:	A factory plans to produce 5,000 TVs in this month. In fact, 60% of the plan is completed in the first half a month, and 70% of the plan is completed in the second half of the month. How many TVs are over-produced this month?
GT Solution:	$5000 \times 60\% + 5000 \times 70\% - 5000$
Teacher (GTS):	$5000 \times (60\% + 70\% - 1)$
Decoder1:	$5000 \times (60\% + 70\%) - 5000$
Decoder2:	$5000 \times 60\% + 5000 \times 70\% - 5000$
TSN-MD:	$5000 \times (60\% + 70\%) - 5000$
Problem Text:	The quotient of a number divided by 54 is 6, and the remainder is 20. What is this number?
GT Solution:	$54 \times 6 + 20$
Teacher (GTS):	$(54 - 20) \div 6$
Decoder1:	$6 \times 54 + 20$
Decoder2:	$6 \times 54 + 20$
TSN-MD:	$6 \times 54 + 20$

Table 5: Generated examples from Math23k dataset.

5 Conclusions

In this paper, we propose a novel approach Teacher-Student Networks with Multiple Decoders (TSN-MD) to tackle the MWP problems. The key components in our approach, i.e., teacher-student regularization and multiple decoders in student network, are validated to be of high effectiveness. The new state-of-the-art results are achieved on the Math23k and MAWPS benchmarks. Furthermore, our approach is actually generic and should be easy to plug in other WMP solvers. Exploring such possibility can be an interesting direction for our future work.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61832001 and No. 61672133), Sichuan Science and Technology Program (No. 2019YFG0535 and No. 2018GZDZX0032), the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative. We thank all reviewers for their valuable comments.

References

[Bobrow, 1964] Daniel G. Bobrow. Natural language input for a computer problem solving system. pages 146–226, 1964.

[Chiang and Chen, 2019] Ting-Rui Chiang and Yun-Nung Chen. Semantically-aligned equation generation for solving and reasoning math word problems. In *NAACL-HLT*, pages 2656–2668, 2019.

[Eigen *et al.*, 2014] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. In *ICLR Workshop Track Proceedings*, 2014.

[Furlanello *et al.*, 2018] Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *ICML*, pages 1602–1611, 2018.

[He *et al.*, 2018] Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. Sequence to sequence mixture model for diverse machine translation. In *CoNLL*, pages 583–592, 2018.

[Hinton *et al.*, 2015] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[Hu *et al.*, 2018] Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. Attention-guided answer distillation for machine reading comprehension. In *EMNLP*, pages 2077–2086, 2018.

[Huang *et al.*, 2017] Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. Learning fine-grained expressions to solve math word problems. In *EMNLP*, pages 805–814, 2017.

[Jacobs *et al.*, 1991] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[Kim and Rush, 2016] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *EMNLP*, pages 1317–1327, 2016.

[Koncel-Kedziorski *et al.*, 2015] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *TACL*, 3:585–597, 2015.

[Koncel-Kedziorski *et al.*, 2016] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In *NAACL*, pages 1152–1157, 2016.

[Kuncoro *et al.*, 2016] Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. Distilling an ensemble of greedy dependency parsers into one MST parser. In *EMNLP*, pages 1744–1753, 2016.

[Li *et al.*, 2019] Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. Modeling intra-relation in math word problems with different functional multi-head attentions. In *ACL*, pages 6162–6167, 2019.

[Liu *et al.*, 2019] Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. Tree-structured decoding for solving math word problems. In *EMNLP-IJCNLP*, pages 2370–2379, November 2019.

[Romero *et al.*, 2015] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[Shen *et al.*, 2019] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. In *ICML*, pages 5719–5728, 2019.

[Shi *et al.*, 2015] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*, pages 1132–1142, 2015.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2017] Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *EMNLP*, pages 845–854, 2017.

[Wang *et al.*, 2018a] Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. Translating a math word problem to a expression tree. In *EMNLP*, pages 1064–1069, 2018.

[Wang *et al.*, 2018b] Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *AAAI*, pages 5545–5552, 2018.

[Wang *et al.*, 2019] Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. Template-based math word problem solvers with recursive neural networks. In *AAAI*, 2019.

[Xie and Sun, 2019] Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305, 7 2019.

[Yang *et al.*, 2018] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *ICLR*, 2018.

[Zagoruyko and Komodakis, 2017] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[Zhang *et al.*, 2018] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, pages 4320–4328, 2018.