Accurate Structured-Text Spotting for Arithmetical Exercise Correction

Yiqing Hu, Yan Zheng, Hao Liu, Deqiang Jiang, Yinsong Liu, Bo Ren
Youtu Lab, Tencent
{hooverhu, neoyzheng, ivanhliu, dqiangjiang, jasonysliu, timren}@tencent.com

Abstract

Correcting arithmetical exercise is a labor intensive and time consuming task for primary school teachers all the time. To reduce their burdens, we propose Arithmetical Exercise Checker (AEC), which is the first system that automatically evaluates all arithmetical expressions (AEs) on exercise images. The major challenge is that AE is formed by printed and handwritten texts with particular arithmetical patterns (*e.g.*, multi-line, fraction). Despite being part of AE, handwritten texts usually lead to zigzag boundaries and tangled rows. What's worse, AE may be arithmetical incorrect, which makes the contextual information less valuable for recognition. To tackle these problems, we introduce integrated detection, recognition and evaluation branches by leveraging AE's intrinsic features, namely 1) *boundary indistinctive*, 2) *locally relevant patterns* and 3) *globally irrelevant symbols*. Experimental results demonstrate that AEC yields a 93.72% correction accuracy on 40 kinds of mainstream primary arithmetical exercises. So far, the online service of AEC processes $75,000$ arbitrary exercises on average per day, and already reduced the burden of over $1,000,000$ users. AEC shows the benefits for implementing an vision-based system as a way to aid teachers in reducing reduplicative tasks.

1 Introduction

With the rapid development of technology, the primary education is evolving all the time. By the end of 2018, there are about a hundred million of primary students in China, and the number is continuing to increase (chE 2018). Now the consensus is that primary teachers should pay more attention to daily performance and learning methodologies of students, which puts forward a higher standard for teachers. However, the time of teachers is still occupied by highly repeated works like correcting exercises. Our survey in 4 primary schools demonstrates that a primary math teacher corrects on average $6,000$ exercises per semester. To tackle this problem, one feasible solution is to have the student parents to correct the homework of their children by their own (which is now a common phenomenon in China), but it requires professional knowledge and not suitable for all parents. Most parents are not familiar with the curriculum and could not give effective guidance to children.

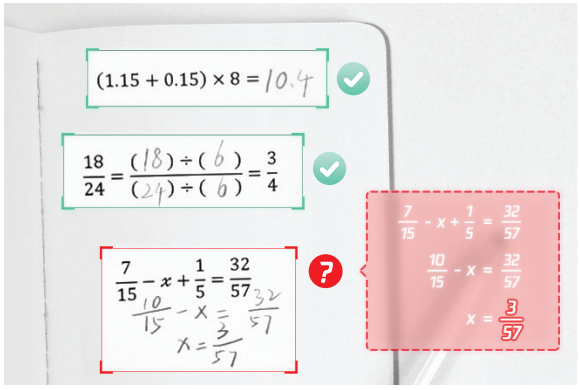

Figure 1: AEC: correcting arithmetical exercise. It localizes, recognizes and evaluates all arithmetical expressions on the exercise, then returns correction results and suggestions.

In this paper, we aim to design such a system that automatically corrects primary arithmetical exercise. To realize this goal, we develop a novel end-to-end approach, arithmetical exercise checker (AEC). It first extracts all AEs by detection, then conducts recognition to convert them to texts. Finally, their correctness is evaluated by arithmetical logics, as shown in Fig. 1. With AEC, the instructors could finish the correction of an exercise within a few seconds, while manual work usually takes a few minutes.

The design of AEC is a non-trivial task. AE is formed by printed and handwritten texts with arithmetical patterns. It has no fixed format according to different publishers and grades, and it could be even purely handwritten. Compared with printed texts, handwritten texts are usually scratchy and lead to 1) zigzag boundaries and 2) tangled rows, as shown in Fig. 2. Zigzag boundaries hamper mainstream anchor-boxes based approaches like SSD (Liu et al. 2016) and Faster-RCNN (Ren et al. 2015), for generated marginal areas are padded by background and lack obvious visual features. What's worse, these approaches tend to set a loose intersection over union (IoU) threshold to balance the number

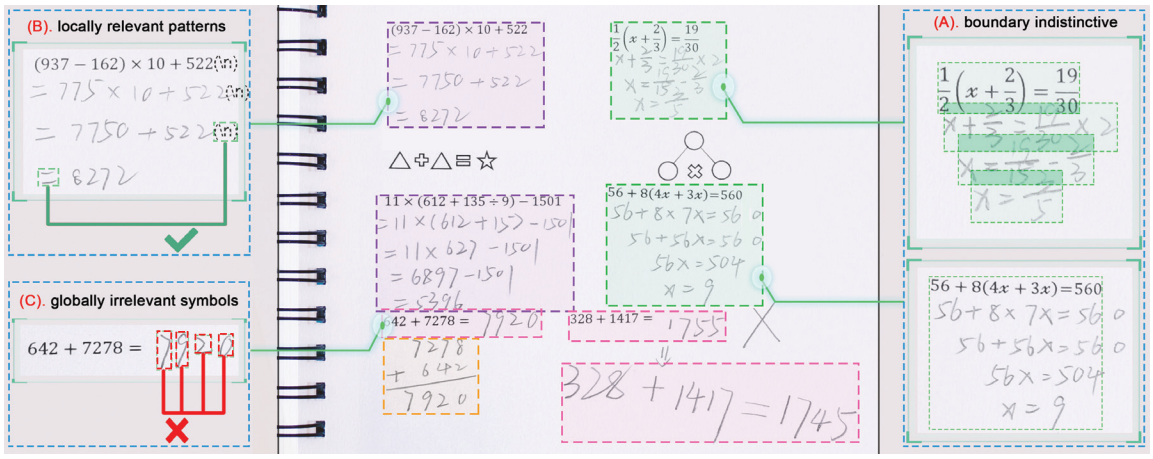

Figure 2: Arithmetical expressions (AEs) are characterized by three distinct features. (A). Tangled rows lead to interference between adjacent lines, and zigzag boundaries hamper mainstream anchor-boxes based approaches. (B). Particular symbols are locally relevant, like "=" usually appears after a "$\backslash n$" (an implicit newline symbol) in formulas. (C). Most symbols have weak or even no semantic relationship with each other, which makes the global context less valuable for inferring individual symbols. Different colors indicate different AE types. Best viewed in color.

between positive and negative proposals. As a consequence, many proposals with cropped AEs are regarded as positive too. On the other hand, tangled rows often lead to interference between adjacent lines, which makes extracting single-line texts a challenging task. Hence, existing text spotting approaches like (Tian et al. 2016) that aim at single-line text suffer in this case. Besides, as AE may be arithmetical incorrect (*e.g.*, $1+1 = 3$), its contextual information is weak. This not only invalids a pre-defined lexicon (Liao et al. 2017), but also makes it harder to train the recognition network.

To overcome these issues, we design the AEC system by exploiting the intrinsic nature of AEs, namely 1) *boundary indistinctive*, 2) *locally relevance patterns* and 3) *globally irrelevance symbols*. We design the AE detection branch based on CenterNet (Duan et al. 2019). It takes convolutional feature as input and outputs a (top-left-corner, center, bottom-right-corner) triplet rather than a rectangular box to represent AE. We introduce an horizontal-focal loss function to let the network pays more attention to learning the harder horizontal boundary. For recognition, we propose an arithmetical neural machine translation (ANMT) approach. ANMT is an encoder-decoder based sequence-to-sequence model, which is inspired by recent progress in automatic caption generation (Xu et al. 2015). Unlike previous text spotting approaches that assume rigid single-line and left-to-right ordering of text, ANMT allows focussing its attention in both horizontal and vertical dimension. Hence, it could cope with structured AEs like multi-line, even with the interference between adjacent lines. Specifically, we design the encoder according to AE's locally relevance pattern and design the decoder by referring to AE's globally irrelevance symbols. The contributions are summarized as follows:

- AEC is the first end-to-end system that automatically corrects arithmetical exercises. We observe and conclude three distinct features of AE, and use them to improve the system design. Experimental results demonstrate the effectiveness of AEC. Now its online service processes $75,000$ arbitrary exercises on average per day, and already reduces the burden of over $1,000,000$ users.
- We first established an arithmetical exercise dataset, AEC-5k. It contains 5300 judiciously selected images, which are collected from 40 mainstream exercises and cover the entire grades of Chinese primary students (usually from age 6 to 12). We will release this dataset soon.

2 Related Works

AEC is essentially a text spotting challenge. In this section, we briefly introduce text spotting approaches and their key components, namely text detection and text recognition.

Text Detection

Existing text detection approaches could be roughly classified into character-based, line-based and word-based methods (Li, Wang, and Shen 2017). With the flourishing of DNN techniques, the word-based methods are popular on account of words are proper targets for mainstream objection detection frameworks (Ren et al. 2015; Liu et al. 2016; Redmon et al. 2016). Inspired by SSD (Liu et al. 2016), TextBoxes (Liao et al. 2017) leverages the combination of different scaled feature maps for text detection. As the text characteristics (*e.g.*, shape, color, *etc.*) are quite different from general objects, many approaches exploit customized region proposal methods. CTPN (Tian et al. 2016) designs a vertical anchor mechanism to jointly predict the location and score for proposals with fixed width.

Beyond the detection of common text, recent approaches focus on more challenging oriented text and twisted text. For oriented text detection, ITN (Wang et al. 2018) encodes the geometric configurations of scene text instances with in-network transformation embedding. In (Lyu et al.

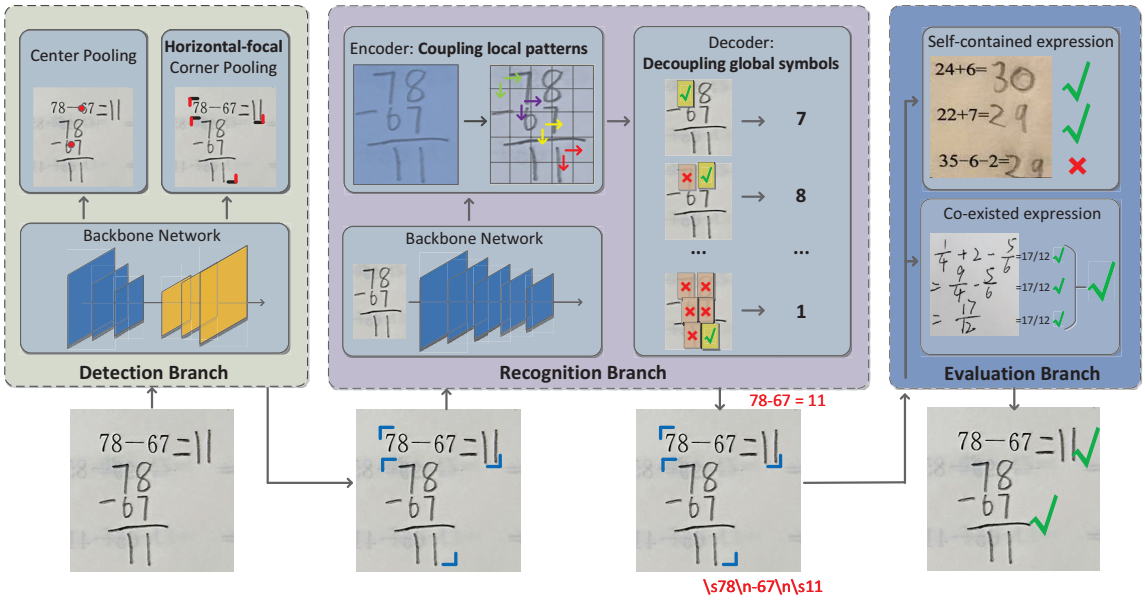

Figure 3: Architecture of AEC. AEC takes an exercise image as input and outputs the evaluation results of all AEs in this image. For the vertical expression "78 - 67 = 11", "$\backslash s$" indicates a blank symbol, and "$\backslash n$" indicates a new line symbol. Colored arrows in "Encoder" indicate that the hidden units exploit two-dimensional feature embeddings. Best viewed in color.

2018), it localizes corner points of text bounding boxes and segmented text regions in relative positions. In (He et al. 2017b), it uses direct regression to predict the text bounding box offset from a given point. For twisted text detection, Textsnake (Long et al. 2018) designs a flexible representation for scene text, which represents text instances in horizontal, oriented and curved forms. Besides, several approaches provide new perspectives in text detection by using Markov clustering network (Liu et al. 2018b) or novel attention mechanism (He et al. 2017a).

Text Recognition

Existing text recognition approaches could be roughly classified into word-level classification-based, sequence-to-label-based and sequence-to-sequence-based methods (Li, Wang, and Shen 2017). Now sequence-to-sequence-based methods are flourishing for their excellent performance in exploiting the context. With the development of RNN techniques, several approaches (Shi, Bai, and Yao 2017; He et al. 2016b) propose deep recurrent models to encode the outputted features of CNN and adopt CTC as text decoder. In (Lee and Osindero 2016; Yang et al. 2017), they utilize an attention-based, sequence-to-sequence framework to focus on specified CNN features in decoding individual characters. For more challenging irregular shaped text, recent work (Shi et al. 2016) exploits a spatial transformer network to rectify the inputted image. These works focus on the recognition of single-line text. Different from these works, AEC is inspired by recent progress in automatic caption generation (Xu et al. 2015) and image-to-markup systems (Deng, Kanervisto, and Rush 2016; Deng et al. 2017), and can recognize structured multi-line texts.

Text Spotting

Existing text spotting approaches consist of two types: 1) using separate detection and recognition models (*a.k.a.* end-to-end at test phase) or 2) using a unified framework (*a.k.a.* end-to-end at both train/test phase). For the first type, it is expected that the separate branches could benefit from more customized feature representations, thus the detector could get precise proposals with tight bounding boxes and the recognizer could generate precise parsing results. Follow this rule, in (Jaderberg et al. 2016), it exploits an ensemble model for detection and a word classifier for recognition. Similarly, the text detection approach TextBoxes (Liao et al. 2017) utilizes CRNN (Shi, Bai, and Yao 2017) for recognition, which helps to prune false positives results generated at the detection phase. For the second type, recent works (Liu et al. 2018a; Li, Wang, and Shen 2017) propose unified neural networks for simultaneous detection and recognition, sharing computation and visual information among the two branches. The unified systems are expected to yield better system efficiency. Our system belongs to the first type.

3 Overall Architecture

Our goal is to 1) design a detection network to localize all AEs, 2) build a recognition network to parse their expressions, and 3) set up the arithmetical logic to evaluate their correctness. The architecture is presented in Fig. 3. The backbone CNN of the detection branch is Hourglass-52 (Newell, Yang, and Deng 2016) and the backbone CNN of the recognition branch is ResNet-50 (He et al. 2016a). We replace the Rectified Linear Unit (ReLU) with the Concatenated ReLU (Shang et al. 2016) in ResNet-50 to reduce the model parameters.

AE Detection Branch

The AE detection network could be mainstream one-stage detectors like SSD (Liu et al. 2016) or two-stage detectors like Faster-RCNN (Ren et al. 2015). Despite their differences, most of them set a number of rectangles with predefined sizes, termed as anchor-boxes, and regress them to ground-truth locations of targets. To ensure the coverage, they usually deploy a large number of anchors with judiciously designed scales and aspect ratios. However, heterogeneous AE types together with the writing style diversity of respondents further increase the difficulty. Facing this problem, we design the detection branch based on CenterNet (Duan et al. 2019). It avoids the usage of anchor-boxes and perfectly fits the *boundary indistinctive* feature of AE.

Boundary indistinctive: We observed that AE is characterized by boundary indistinctive. Compared with general objects, boundaries of letters/digits are implicit because they are usually hollowed with concaved edges. If using rectangular bounding boxes to represent AEs, their marginal areas may be padded by background and lack obvious visual features. Besides, the bounding boxes generated by anchor-boxes based approaches are not precise enough. The insight is that these approaches tend to set a loose intersection over union (IoU) threshold for positive samples, to balance the number between positive and negative samples and thus facilitating the training process. Taking a common threshold IoU $= 0.7$, many samples with imprecise bounding boxes are denoted as positive samples too.

Facing these problems, we design our AE detection network. It takes convolutional features as input and outputs triplets rather than rectangular boxes to represent AEs. The triplet consists of 1) left-up corner keypoint, 2) center keypoint and 3) right-bottom corner keypoint. The corner keypoints pair is used to generate the proposal and the center keypoint is used to confirm its validity. The visual patterns of the triplet are extracted by cascade corner pooling and center pooling defined in CenterNet, respectively. We suggest to refer to (Duan et al. 2019) for details. Compared with anchor-boxes based approaches, our detection network better extracts feature of AE corners and helps to generate complete proposals.

For better pinpointing corners of AEs, we propose using a horizontal-focal loss function for corner regression. Compared with its counterpart in CenterNet, it imposes harsher penalties against corner shift on the horizontal axis. The intuition is that it is harder to learn the horizontal boundaries compared with the vertical boundaries. If one AE loses part of its expression horizontally, it could still be a valid expression with a high probability (dotted blue rectangle), which does not apply in the vertical case (red rectangle), as shown in Fig. 4. Hence, the horizontal boundary feature of AE is more indistinctive and is harder to be learned. In CenterNet, the loss of a predicted corner is defined as $L_{det} \propto -e^{-(\Delta x^2+\Delta y^2)}$, where Δx and Δy are coordinate shifts of the predicted corner, according to the ground-truth. Basing on this observation, we propose the horizontal-focal loss function as:

$$L_{det} \propto -e^{-(\alpha\Delta x^2+\Delta y^2)} \tag{1}$$

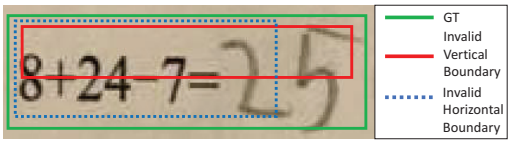

Figure 4: We propose the horizontal-focal loss function to pinpoint the harder horizontal boundaries. The detection network rarely generates proposals with invalid vertical boundaries (red rectangle), but often generates proposals with invalid horizontal boundaries (dotted blue rectangle). "GT" stands for the ground truth rectangular box for this AE.

We use α to control the penalty, which is set default as 2 experimentally. With this loss function, our detection branch better pinpoints the challenge horizontal boundaries of AEs.

AE Recognition Branch

Most existing text-spotting approaches focus on the recognition of single-line text (Shi, Bai, and Yao 2017; He et al. 2016b; Lee and Osindero 2016). However, they are not suitable for recognizing AEs, for they are usually structured, scratchy handwritten and multi-line texts. Facing this problem, we design an arithmetical neural machine translation module (ANMT) for recognition. ANMT is an encoder-decoder based sequence-to-sequence model, which is inspired by recent progress in automatic caption generation (Xu et al. 2015; Sutskever, Vinyals, and Le 2014). Unlike approaches that assume rigid single-line and left-to-right ordering of text (1D), ANMT allows to focus its attention in both horizontal and vertical dimension of AEs (2D). Besides, we observe that AEs have another two distinct features, namely *locally relevant patterns* and *globally irrelevant symbols*. We enhance the recognition branch by leveraging these features.

ANMT bases on the original NMT framework (Bahdanau, Cho, and Bengio 2014). It models the translation probability from CNN's feature embedding $\mathbf{x}$ to expression $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$. Note that the original NMT denotes CNN's feature embedding as $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ and permuted it to 1D time major form $\{x_1, \cdots, x_i, \cdots, x_W\}$, where $x_i \in \mathbb{R}^{C\times H}$ is a vertical clip from the feature map with channel C, width W and height H. This embedding fails to represent AEs with multi-line texts. Inspired by (Deng, Kanervisto, and Rush 2016; Deng et al. 2017), we keep the feature map $\mathbf{x} = \{x_{11}, \cdots, x_{ij}, \cdots, x_{HW}\}$ unchanged, where $x_{ij} \in \mathbb{R}^C$. Thus we have:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \prod_{t=1}^{m} P\left(y_t|\mathbf{y}_{<\mathbf{t}}, \mathbf{x}; \theta\right) \\ &= \prod_{t=1}^{m} \text{softmax}(f(y_{t-1}, s_t, c_t)) \end{aligned} \tag{2}$$

where $f(\cdot)$ is a nonlinear function that outputs the probability of y_t according to the previous output y_{t-1}, current hidden state s_t and context vector c_t. For decoder, we select uni-directional LSTM $g(\cdot)$ as the function, which is formed

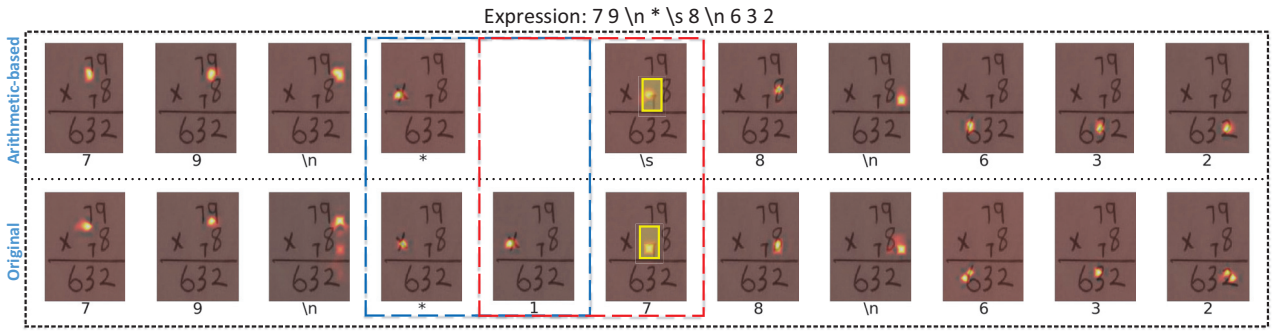

Figure 5: Comparison between the arithmetical attention and the original attention in recognizing a vertical expression. "$\backslash s$" indicates a blank symbol, and "$\backslash n$" indicates a new line symbol. Our method forces the visual embedding of "$*$" to be attended only once during the entire decoding process (the dotted blue rectangle), which avoids the error recognition of "1". Our method also helps to refine the hidden states and ignores the tiny "7" (yellow rectangle). It is a carry indicator that often appears in handwritten vertical expressions, which is considered to be a severe noise. Best viewed in color.

by recurrent s_t:

$$s_t = g(s_{t-1}, y_{t-1}, c_t) \tag{3}$$

where c_t shows the attention on different hidden states h_{ij} in the encoder:

$$c_t = \sum_{i=1}^{H} \sum_{j=1}^{W} a_{tij} h_{ij} \tag{4}$$

The attention weight a_{tij} is calculated as follows:

$$a_{tij} = \frac{\exp(e_{tij})}{\sum_{k=1}^{H} \sum_{l=1}^{W} \exp(e_{tkl})} \tag{5}$$

where e_{tij} scores how much the hidden state s_{t-1} in decoder attends to hidden state h_{ij} in the encoder. Herein e_{tij} has several definitions (Luong, Pham, and Manning 2015), and we adapt $e_{tij} = \mathbf{v}_a^T \tanh(\mathbf{W}_a s_{t-1} + \mathbf{U}_a h_{ij})$ here.

Finally, the training objective of ANMT is to maximize the log-likelihood of the training instances $(\mathbf{x}^s, \mathbf{y}^s)$:

$$\theta^* = \arg\max_{\theta} \sum_{s=1}^{S} \log p(\mathbf{y}^s | \mathbf{x}^s) \tag{6}$$

Local relevance: coupling arithmetical patterns: AEs are characterized by the local relevance between particular symbols. For instance, an "=" usually appears after a "$\backslash n$" in formulas. These particular local patterns could be encoded in the hidden state to amplify the contextual information. To realize this goal, we use MD-LSTM (Voigtlaender, Doetsch, and Ney 2016) for encoding. Compared with the commonly used bi-LSTM, MD-LSTM encodes the feature embedding along both axes and produces a transformed embedding of the same size. It allows the encoder to see more context and leads to more stable training. Besides, trainable initial hidden states are inserted at the start position of each row as vertical position embeddings, which are utilized to capture the sequential order information in the vertical direction.

Global irrelevance: decoupling arithmetical symbols: Another distinct feature of AE is that it may be arithmetically incorrect (*e.g.*, $1 + 1 = 3$). At the training phase, the arithmetical expression acts as the target context $\mathbf{y}$ in Eqn. (3). Except for a few arithmetical patterns, most symbols (*e.g.*, digits, letters) are randomly written by respondents. Thus, these symbols are globally irrelevant and have weak or even no semantic relationship with each other. This makes the target context less valuable for the decoder. Motivated by this observation, we use a context gate to let the decoder to focus more on source contexts, *a.k.a.* the embedding of visual features. This context gate is essentially a gate function that dynamically controls the ratios at which source and target contexts contribute to the generation of target words (Tu et al. 2017). Formally, we have:

$$w_t = \sigma(s_{t-1}, y_{t-1}, c_t) \tag{7}$$

where w_t denotes the context weight, $\sigma(\cdot)$ is a logistic sigmoid function. With w_t, the original decoding process in Eqn. (3) is replaced by:

$$s_t = g(s_{t-1}, y_{t-1}, w_t \cdot c_t) \tag{8}$$

AE's globally irrelevant feature further guides the design of attention mechanism. As shown in Eqn. (4), the attention mechanism extracts source contexts by aggregating all hidden states. Since symbols have weak or even no semantic relationship with each other, a hidden state should only obtain a significant weight once during the entire decoding process. This significant weight is achieved when the decoder is inferring the hidden state's presented symbol if existed. For example, the visual feature embedding of symbol "$*$" is attended when inferring "$*$", and it should not be heavily attended when inferring other symbols, as shown in Fig. 5. To realize this goal, we keep a mask matrix $\mathbf{M}$ whose size is the same as $\mathbf{h}$, where the default value of all symbols is 1. For a specific element $h_{ij} \in \mathbf{h}$, if its corresponding weight $a_{tij} > \gamma$, where γ is cut-off threshold and is set default as 0.5, we have:

$$m_{ij} = 0, \text{ s.t. } a_{tij} > \gamma \tag{9}$$

Hence, Eqn. (4) is replaced by:

$$c_t = \sum_{i=1}^{H} \sum_{j=1}^{W} a_{tij} \cdot h_{ij} \cdot m_{ij} \tag{10}$$

Beyond the recognition branch, we also exploit candidates of AE to improve the recognition. Note that AEs are usually formed by dozens of symbols, and a single recognition error may bias the correction result with a high probability. Facing this problem, we generate multiple candidates of AE by slight rotation, resizing or padding. These candidates are recognized together with the original AE. The majority expression in the results is selected as the output, or expression of the original AE is selected if no majority exists.

AE Evaluation Branch

AEs could be essentially divided into two classes: self-contained and co-existed expressions. For the self-contained expression, its correctness could be evaluated by itself. This type of expression could be denoted as $f \bigotimes g$, where f and g are combinations of basic arithmetical operations like $C_0 \times (C_1 - C_2) \div C_3$, herein C_0, C_1, C_2, C_3 are rational numbers and $\bigotimes$ denotes a operation symbol such as "=". AEC evaluates f and g respectively first and then checks the correctness of $f \bigotimes g$. For the co-existed expression (*e.g.*, a formula), AEC evaluates all sub-expressions and checks their consistency. For instance, a single formula contains the question $f(x) = C_0$ (*e.g.*, $2x - \frac{1}{2} = \frac{3}{2}$), an intermediate function $g(x) = C_1$ (*e.g.*, $2x = 2$) and the answer $x = C_2$ (*e.g.*, $x = 1$). AEC evaluates the question and all intermediate functions and gets consistent results $x = C_2$, which proves the correctness of this formula. Otherwise, AEC returns false. If so, AEC will also provide suggestions to respondents according to the error type, as shown in Fig. 1.

4 Experiments

Datasets: Now there exist no public datasets of arithmetical exercises. Hence, we construct the AEC-5k dataset, which is formed by 40 kinds of mainstream primary exercises. AEC-5k consists of $5,000$ images for training and 300 images for testing, with mean resolution 1152×768 and average 8.7 AEs in each image. each AE annotation has two attributes: 1) a bounding box that covers the entire AE even if it is multi-lined, and 2) a char-level text annotation. 120 kinds of different characters appear in annotations, which comprise of numbers (*e.g.*, "1", "2"), operation symbols (*e.g.*, "+", "×"), uppercase/lowercase English letters (*e.g.*, "cm", "kg") and their mappings in Chinese. As the labeled data is insufficient for training the recognition branch, we synthesize a 600k handwritten corpus by referring to (Deng et al. 2017). We'll release these datasets soon.

Criteria: we adapt two metrics for evaluation: 1) the evaluation protocols of ICDAR 2015 (Karatzas et al. 2015) that measure the text-spotting performance. As AEs are permutations of arbitrary symbols, we select the general "End-to-End" protocol for it works without a contextualized lexicon. 2) the AE correction accuracy to measure the performance of AEC. An AE is considered to be "correct" if its expression satisfies the corresponding arithmetical logics.

Implementation Details

Basing on Pytorch (Paszke et al. 2017), we implement all benchmarks on a regular platform with 8 Nvidia P40 GPUs and 64GB memory. We initialize the detection branch without pretraining on any external dataset. The AEC-5k training data is used to fine-tune the model until convergence. The training data is augmented by: 1) randomly rescaling the width of the image by ratio range $[0.8, 1.2]$ without changing its height, and 2) randomly extracting image crops from the original image with ratio range $[0.75, 1]$; Image crops with truncated AEs are discarded. We adapt the Adam optimizer with learning rate 2.5×10^{-4} for optimization. We apply NMS (Rosenfeld and Thurston 1971) on generated text regions. For the recognition branch, we use the AEC-5k training data and the 600k synthetic data for training without a pre-trained model. We adapt the SGD optimizer with learning rate 0.1 for optimization. The learning rate halves after 300k iterations, and halves again after each 100k iterations.

AEC: Ablation study

We conduct benchmarks to investigate the contribution of proposed components in the AEC system.

Boundary indistinctive: Motivated by the boundary indistinctive feature, AEC detection branch uses the horizontal-focal loss function for corner regression. We observe that by replacing the original loss in CenterNet with the horizontal-focal loss ($\alpha = 2$), the precision, recall, and F-measure are increased by 1.77%, 0.64% and 1.23% respectively. $\alpha = 2$ is also a balanced trade-off for learning both the horizontal and the vertical boundary, As shown in Table 1. Additionally, we use multi-scale input for improvement. In the multi-scale setting, resolution of images is set as $\{0.8, 1, 1.2\}$ times referring to the original image. The multi-scale input produces a more robust result, and we select "AEC detection (MS, $\alpha = 2$)" as the default detection approach.

Locally relevant patterns: In the method "ANMT w/o local."in Table 2, we replace the original bi-LSTM encoder in ANMT with an MD-LSTM encoder, to embed the contextual information of locally relevant arithmetical patterns. We observe that the recognition accuracy and the correction accuracy are increased by 0.52% and 0.68% respectively. It validates that the MD-LSTM encoder helps to encode arithmetical patterns.

Globally irrelevant symbols: It guides the design of two modules in ANMT's decoder: 1) the context gate and 2) the arithmetical attention mechanism. In the method "ANMT w/o cg.", we remove the context gate. As a consequence, the recognition accuracy and the correction accuracy are decreased by 0.75% and 1.40% respectively. In the method "ANMT w/o attn.", we replace the arithmetical attention module with the default global attention. This operation decreases the recognition accuracy and the correction accuracy by 0.88% and 1.58% respectively.

Besides, we exploit AE's candidates for improvement. Given an AE, we generate its candidates by slight rotation, resizing or padding. Two candidates are generated for each action. For resized candidates ("ANMT with res."), the resolution range is [0.8, 1.2]; For padded candidates ("ANMT

Detection Method	Precision	Recall	F
CenterNet ($\alpha = 1$)	89.61	94.13	91.79
AEC detection ($\alpha = 3$)	88.31	90.43	89.35
AEC detection ($\alpha = 5$)	79.52	83.04	81.24
AEC detection ($\alpha = 2$)	**91.38**	**94.77**	**93.04**
AEC detection (MS, $\alpha = 2$**)**	**94.55**	**97.02**	**95.76**

Table 1: Ablation study of the AEC detection branch. "F" represents F-measure in percentage, and "MS" stands for multi-scale.

Recognition Method	End-to-End	Accuracy
ANMT w/o local.	91.71	90.43
ANMT w/o cg.	91.48	90.71
ANMT w/o attn.	91.35	90.53
ANMT	92.23	91.11
ANMT with res.	93.39	92.16
ANMT with pad.	93.22	91.83
ANMT with rot.	93.25	91.84
ANMT (AC)	**94.32**	**93.72**

Table 2: Ablation study of the AEC recognition branch. "w/o local.", "w/o cg." and "w/o attn." are short for "without using the MD-LSTM", "without using the context gate" and "without using the arithmetical attention", respectively. "w/o res.", "w/o pad." and "w/o rot." stands for without resized, padded and rotated candidates, respectively. "AC" stands for using candidates generated by all actions. "End-to-End" indicates the general ICDAR 15 protocol without a lexicon. "Accuracy" stands for the AE correction accuracy.

with pad."), the ratio for blank paddings is [0.05, 0.25]. For rotated candidates ("ANMT with rot."), the angle is limited within [-10°, 10°]; In the method "ANMT (AC)", all candidates generated by three actions are used. We use batch operations for acceleration. Compared with ANMT, "ANMT (AC)" has a 2.09% recognition improvement and a 2.61% correction accuracy improvement, at the cost of an additional 27% inference speed. We select "ANMT (AC)" as the default recognition approach, for users are more sensitive to the correction accuracy, as reported in the user feedback.

AEC: Comparison with the State-of-the-Art

Detection: We train the detection branch with AEC-5k training data and its augmentation. The same data is used to implement training of mainstream one-stage and two-stage text detection approaches. For anchor-box based approaches, we set 6×12 anchors according to the K-means clustering result on aspects and ratios of the AE objects in the training data. The AEC-5k testing data is used for evaluation. Experimental results demonstrate that our method yields more precise proposals compared with anchor-box based approaches, which are crucial for the following AE recognition.

Recognition and correctness evaluation: We train the recognition branch with AEC-5k training data and the 600k synthetic data. The same data is used to implement recent text recognition approaches. To evaluate the end-to-end metric together with the correction accuracy, we use the proposal generated by "AEC detection (MS)" as the input for all recognition approaches. Note that CRNN (Shi, Bai, and Yao 2017) targets at the single-line text, thus it can not recognize multi-line AEs directly. For a fair comparison, we add extra single-line text vertex annotations on AEC-5k training data, then build a SSD (Liu et al. 2016) model with customized anchors to extract single-line texts. These texts are recognized separately, and the outputted expressions are concatenated together as the results. Compared with these approaches, our approach achieves the best recognition performance and correction accuracy.

Detection Method	Precision	Recall	F
TextBoxes (Liao et al. 2017)	84.75	84.00	84.37
EAST (Zhou et al. 2017)	85.10	85.45	85.28
RPN (Ren et al. 2015)	54.32	64.25	58.87
Faster-RCNN (Ren et al. 2015)	73.91	93.03	82.38
Faster-RCNN + OHEM	74.82	94.12	83.37
Faster-RCNN + OHEM (MS)	88.77	93.05	90.86
AEC detection	**91.38**	**94.77**	**93.04**
AEC detection (MS)	**94.55**	**97.02**	**95.76**

Table 3: Comparison with state-of-the-arts text detection approaches.

Text-spotting Method	End-to-End	Accuracy
CRNN (Shi, Bai, and Yao 2017)	65.91	62.12
Im2markup	86.36	83.60
ANMT	92.23	91.11
ANMT (AC)	**94.32**	**93.72**

Table 4: Comparison with state-of-the-arts text-spotting approaches. Im2markup is proposed in (Deng, Kanervisto, and Rush 2016).

AEC: User Feedback

Through an on-line feedback page, we received 1643 valid comments across 338 days. Most participants consider AEC a beneficial tool by comments such as: "It could calculate multiple formulas simultaneously, and it really saves time", and, "The correction speed is promising, really helpful". At the same time, some feedbacks also point out the deficiencies in the current system. For example, one participant said, "it could not evaluate the graphic computation (e.g., $\Delta + \Delta = 2\Delta$)". We will digest their suggestions to improve the system design in the future.

5 Conclusion

In this paper, we proposed AEC, an end-to-end system that automatically corrects primary arithmetical exercises. The design of AEC roots in three distinct features of arithmetical expressions, which are obstacles for existing text-spotting approaches. We also introduced the AEC-5k dataset to assist the study of AEC, which consists of $5,300$ images from 40 mainstream primary exercises. Future directions would be to support the evaluation of more expression types.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

2018. http://www.moe.gov.cn/.

Deng, Y.; Kanervisto, A.; Ling, J.; and Rush, A. M. 2017. Image-to-markup generation with coarse-to-fine attention. In *ICML*, 980–989. JMLR. org.

Deng, Y.; Kanervisto, A.; and Rush, A. M. 2016. What you get is what you see: A visual markup decompiler. *arXiv preprint arXiv* 1609.

Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; and Tian, Q. 2019. Centernet: Object detection with keypoint triplets. *arXiv preprint arXiv:1904.08189*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *CVPR*, 770–778.

He, P.; Huang, W.; Qiao, Y.; Loy, C. C.; and Tang, X. 2016b. Reading scene text in deep convolutional sequences. In *AAAI*, 3501–3508.

He, P.; Huang, W.; He, T.; Zhu, Q.; Qiao, Y.; and Li, X. 2017a. Single shot text detector with regional attention. In *ICCV*, volume 6.

He, W.; Zhang, X.-Y.; Yin, F.; and Liu, C.-L. 2017b. Deep direct regression for multi-oriented scene text detection. In *ICCV*, 745–753.

Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2016. Reading text in the wild with convolutional neural networks. *IJCV* 116(1):1–20.

Karatzas, D.; Gomez-Bigorda, L.; Nicolaou; et al. 2015. Icdar 2015 competition on robust reading. In *ICDAR*, 1156–1160. IEEE.

Lee, C.-Y., and Osindero, S. 2016. Recursive recurrent nets with attention modeling for ocr in the wild. In *CVPR*, 2231–2239.

Li, H.; Wang, P.; and Shen, C. 2017. Towards end-to-end text spotting with convolutional recurrent neural networks. In *ICCV*, 5238–5246.

Liao, M.; Shi, B.; Bai, X.; Wang, X.; and Liu, W. 2017. Textboxes: A fast text detector with a single deep neural network. In *AAAI*, 4161–4167.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *ECCV*, 21–37. Springer.

Liu, X.; Liang, D.; Yan, S.; Chen, D.; Qiao, Y.; and Yan, J. 2018a. Fots: Fast oriented text spotting with a unified network. In *CVPR*, 5676–5685.

Liu, Z.; Lin, G.; Yang, S.; Feng, J.; Lin, W.; and Ling Goh, W. 2018b. Learning markov clustering networks for scene text detection. In *CVPR*, 6936–6944.

Long, S.; Ruan, J.; Zhang, W.; He, X.; Wu, W.; and Yao, C. 2018. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *ECCV*, 19–35. Springer.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, 1412–1421.

Lyu, P.; Yao, C.; Wu, W.; Yan, S.; and Bai, X. 2018. Multi-oriented scene text detection via corner localization and region segmentation. In *CVPR*, 7553–7563.

Newell, A.; Yang, K.; and Deng, J. 2016. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, 483–499. Springer.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *CVPR*, 779–788.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 91–99.

Rosenfeld, A., and Thurston, M. 1971. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers* (5):562–569.

Shang, W.; Sohn, K.; Almeida, D.; and Lee, H. 2016. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2217–2225.

Shi, B.; Bai, X.; and Yao, C. 2017. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *TPAMI* 39(11):2298–2304.

Shi, B.; Wang, X.; Lyu, P.; Yao, C.; and Bai, X. 2016. Robust scene text recognition with automatic rectification. In *CVPR*, 4168–4176.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*, 3104–3112.

Tian, Z.; Huang, W.; He, T.; He, P.; and Qiao, Y. 2016. Detecting text in natural image with connectionist text proposal network. In *ECCV*, 56–72.

Tu, Z.; Liu, Y.; Lu, Z.; Liu, X.; and Li, H. 2017. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics* 5:87–99.

Voigtlaender, P.; Doetsch, P.; and Ney, H. 2016. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 228–233. IEEE.

Wang, F.; Zhao, L.; Li, X.; Wang, X.; and Tao, D. 2018. Geometry-aware scene text detection with instance transformation network. In *CVPR*, 1381–1389.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2048–2057.

Yang, X.; He, D.; Zhou, Z.; Kifer, D.; and Giles, C. L. 2017. Learning to read irregular text with attention mechanisms. In *IJCAI*, 3280–3286.

Zhou, X.; Yao, C.; Wen, H.; Wang, Y.; Zhou, S.; He, W.; and Liang, J. 2017. East: an efficient and accurate scene text detector. In *CVPR*, 2642–2651. IEEE.