Neural Computing and Applications
https://doi.org/10.1007/s00521-023-08890-6

ORIGINAL ARTICLE

FATE: a three-stage method for arithmetical exercise correction

Qipeng Zhu[1,4] · Zhuoyan Luo[1,4] · Shipeng Zhu[1,2] · Qi Jing[3,4] · Zihang Xu[3,4] · Hui Xue[1,2]

Received: 1 November 2022 / Accepted: 12 July 2023

Abstract
As the number of primary students rapidly rises, the highly repetitive task of correcting arithmetical exercises consumes much time for teachers and hinders them from concentrating more on the growth of students. To reduce the workload of teachers, arithmetical exercise correction (AEC) is proposed to automatically detect, recognize and correct various arithmetical exercises in the workbook. However, two crucial issues need to be addressed since the research in this field is still immature, i.e., accurate detection of the arithmetic exercise with various structures and the effective recognition of long-size exercise. In this paper, we propose a three-stage method dubbed as FATE, to correct arithmetical exercises in an end-to-end manner. Specifically, we apply the anchor-free model with a feature pyramid network and constraint of centerness to avoid the redundant bounding boxes. On the other hand, we employ a transformer-based framework with contrastive learning to extract global symbol information and generate corresponding sequences. Finally, we design a series of rule-based templates to correct the generated sequence based on the unique features of each type of arithmetical exercises, respectively. Extensive experiments demonstrate that our method yields the detection average precision of 96.8%, the recognition accuracy of 92.3% and the F_1 score of 91.2% in spotting experiment on the public dataset, which outperforms the state-of-the-art method.

Keywords Arithmetical exercise · Transformer · Contrastive learning · Spotting

1 Introduction

To develop the basic logic and mathematical skills of primary students, arithmetical exercises contain a large number of reasoning and calculations based on clear rules, e.g., fractional and approximate equations. However, correcting these repetitive exercises is labor-intensive and time-consuming, which occupies much time for teachers. Arithmetical exercise correction (AEC) [1] aims to automatically spot and correct the arithmetical sequences from the exercise image. It consists of three sub-tasks: (1) Detecting regions containing arithmetical exercises in images. (2) Recognizing arithmetical exercises from the detected proposals and generating corresponding mathematical character sequences. (3) Correcting the generated exercise sequences. With the diversification of education scenarios and content, the AEC models reduce the burdens of teachers and help them pay more attention to the overall development of students. Furthermore, it plays a key role in online education.

Arithmetical exercise correction is a comprehensive task with more challenges than traditional tasks. There are three main difficulties in arithmetical exercise correction: (1) Diversified structures prevent the localization in the detection branch. For example, as shown in Fig. 1c, the multiple short equations have similar structures with the sub-equations of the recursive equation, which may mislead the conventional models to produce redundant proposals. (2) Complicated expressions bring challenges to the recognition branch. That is, the nested relations of sub-

Qipeng Zhu, Zhuoyan Luo, and Shipeng Zhu have contributed equally to this work.

✉ Hui Xue
hxue@seu.edu.cn

1 School of Computer Science and Engineering, Southeast University, Nanjing 211189, Jiangsu, China

2 Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, Nanjing 211189, Jiangsu, China

3 Chien-Shiung Wu College, Southeast University, Nanjing 211189, Jiangsu, China

4 Computer Experimental Teaching Center of Southeast University, Nanjing 211189, Jiangsu, China

Published online: 05 September 2023

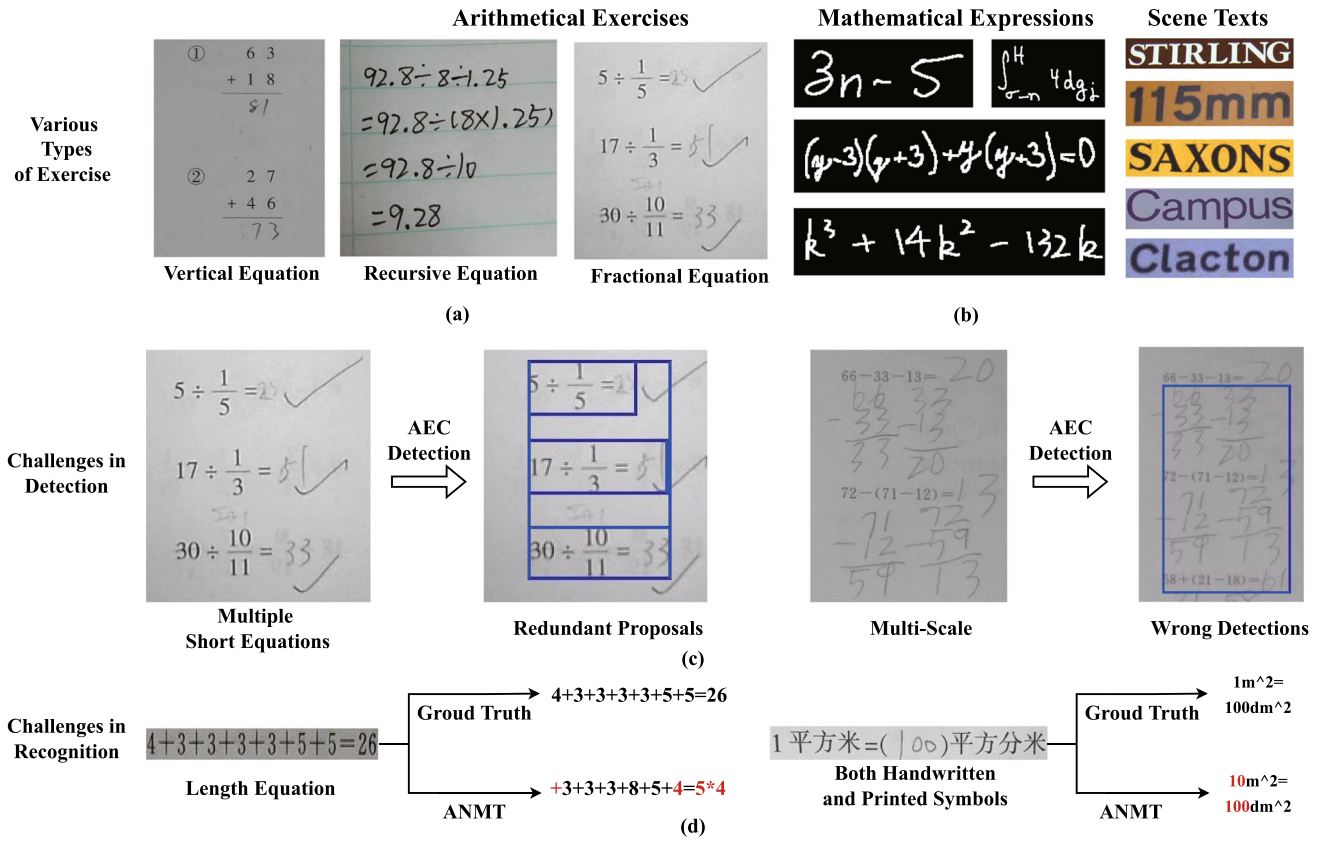

Fig. 1 The sub-figure (**a**) presents representative types of arithmetical exercises, while the sub-figure (**b**) demonstrates Latex-based mathematical expressions. The sub-figure (**c**) and sub-figure (**d**) depict the challenges in detection and recognition

equations (shown in Fig. 1a) make the model need to consider global information. Meanwhile, the image quality and clarity of handwritten symbols put forward requirements for robustness. (3) Correction branch relies on the above two branches, which put forward high requirements for the performance of the detection and recognition branches. Meanwhile, the rich forms of expressions make it difficult to pre-store and compare answers to all expressions. Notably, there are two related tasks for the arithmetical exercise correction, i.e., scene text spotting and handwritten mathematical expression recognition. Compared to them, arithmetical exercises have multi-domain symbols, i.e., the mixture of printed and handwritten symbols, and complex spatial structures, while the counterparts have one-domain and one-line forms, respectively.

Previous handwriting mathematical expression recognition (HMER) work [2–4] focuses on Latex-based expression and achieves notable performance. However, there are two key differences between arithmetical exercises and Latex-based mathematical expression: (1) Arithmetical exercises consist of multi-scale structures (see in Fig. 1a) due to the wide variety of exercises, e.g., the recursive equation and the fractional equation. In contrast, the majority of Latex-based mathematical expressions have only one-line structures. (2) Arithmetical exercises include both printed and handwritten text, whereas Latex-based ones are entirely made up of handwritten characters (see in Fig. 1b). Moreover, the correction procedure relies on the preceding steps since it can be implemented by rule-based algorithms based on the generated arithmetical sequence.

To achieve the automatic correction of arithmetical exercises, Hu et al. [1] proposed arithmetical exercise checker, which localizes and crops arithmetical exercise regions on the input image, then recognizes the sequence of the arithmetical exercise by feeding the cropped images into its recognition branch. Since this work is the first model to address the AEC task, many problems remain to be improved. The original detection branch, CenterNet, struggles to suppress low-quality anchor boxes validly. In addition, it often fails to predict the scale of the arithmetical exercise precisely (see Fig. 1c). The recognition branch, arithmetical neural machine translation (ANMT), produces low-precision results without synthetic data. It fails when images contain lengthy arithmetical exercises since the decoder of ANMT hardly captures dependency information as shown in Fig. 1d. Meanwhile, this method requires a lot of manually labeled data, which are scarce and expensive due to the complexity of mathematical labeling.

To address the aforementioned issues, we propose a three-stage automatic correction method called FATE for primary school student arithmetical exercises. It first *F*ocuses on the region containing the arithmetical exercise and attaches more *AT*tention to generate corresponding arithmetical sequences, then corr*E*ct each exercise based on rules. Specifically, the detection branch adopts a feature pyramid network (FPN) [5] network for accurate multi-level arithmetical exercises detection. Furthermore, the center-ness scheme [6] is introduced to reduce redundant proposals. For the recognition branch, considering the appearance of crabbed text and the high similarity among arithmetical characters, we employ the encoder to extract the high-level difference of features of proposals. Meanwhile, we apply the contrastive learning method to enhance the representation capabilities of the encoder in order to tackle the intricate and various structures of mathematical problems. Then, a transformer-based decoder is designed to establish the global semantic dependency, which drastically exploits the long-distance correlation of arithmetical exercises. Finally, rule-based templates are designed to correct each type of arithmetical exercise.

In conclusion, our main contributions to the paper are as follows:

- We propose an effective end-to-end three-stage method to automatically correct arithmetical exercises for primary school students.
- We employ the anchor-free detection model with FPN [5] and center-ness [6] to locate each arithmetical exercise in images. Then, we adopt the transformer-based encoder–decoder framework with a contrastive learning paradigm for exercise recognition. Moreover, we design rule-based templates to correct arithmetical assignments.
- Comprehensive experiments on the AEC-5k dataset without synthetic data show that our model outperforms state-of-the-art methods in both detection and recognition, proving its superior performance.

2 Related work

2.1 Arithmetical exercise detection

Limited by the insufficient handwritten data, researchers first focus on the detection of printed mathematical expressions. Ohyama et al. [7] first propose a U-Net-based method to localize the printed mathematical expressions in document images. Then, ScanSSD [8] introduces the voting-based pooling into general SSD [9] and achieves promising performance. In [1], authors first propose the new task: arithmetical exercise correction. They apply the CenterNet [10] to localize the exercises, which shows the effectiveness of the general object detection methods. To accurately locate the scene text in the natural environment, multi-oriented text detection methods [11–13] exists. For instance, text-block fully convolutional network (FCN) [11] takes into account both local and global cues for localizing text lines in a coarse-to-fine procedure. On the other hand, general object detection methods can be classified into two categories: anchor-based and anchor-free methods. Most of the anchor-based detection methods [14–17] need to pre-define many anchor boxes such as Faster-RCNN [15], YOLOv3 [16] and YOLOf [18]. In contrast, anchor-free methods [6, 10] tend to directly output the detection bounding boxes. Notably, the shapes of arithmetical exercises are diversified and similar to the objects for general detection methods, e.g., vertical and fractional equations. Therefore, text detection methods struggle to address this problem. Moreover, the pre-defined anchors from anchor-based methods limit the detection performance for objects with large-scale differences. However, anchor-free methods use suitable modules to achieve high accuracy and recall of detection.

2.2 Arithmetical exercises recognition

The handwritten mathematical expression recognition, leading to a board downstream applications, has been extensively studied in the community. Nowadays, sequence-to-sequence-based methods are thriving due to their outstanding performance in establishing context dependency. Encoder–decoder framework, one of the sequence-to-sequence methods, is widely used in mathematical expression recognition. In the existing method, WAP [19] applies an FCN as the encoder and first introduces the 2D coverage to solve the problem. The model proposed in [4], uses two inverse decoders learning mutually to enhance the paring ability of decoders, which attains high accuracy. Furthermore, CAN [2] introduces a symbol counting technique in mathematical expression recognition, which provides global information while decoding the expression. Nevertheless, the mathematical expression and arithmetical exercise have certain similarities but the structure of the arithmetical exercise is more diversified, and both printed and handwritten symbols occur alternately. ANMT, proposed in [1], is a 2D attention-based encoder–decoder framework, which uses ResNet-50 [20] and MDLSTM [21] as the encoder for feature extraction, while LSTM [22] is used as the decoder to recognize arithmetical exercises. For the extended arithmetical exercises, BTTR [3] utilizes transformer decoder [23] to effectively capture the attention for accurate recognition.

2.3 Text spotting

Enlightened by the close relationship between detection and recognition, some researchers commenced constructing a unified model to achieve end-to-end training. It is anticipated that the detection and recognition module share features to learn reciprocally. Li et al. [24] first propose an end-to-end method, but it takes a long time to process. Xuebo Liu et al. propose an end-to-end model fast oriented text spotting (FOTS) [25] based on bidirectional LSTM and connectionist temporal classification (CTC), which reaches a relatively high speed both training and inference. For structured text, especially arithmetical exercises, AEC [1] puts forward a two-stage method for spotting, where it adopts CenterNet to detect the region covering arithmetical exercises and employs ANMT to generate the sequence.

2.4 Contrastive learning

Contrastive learning is a typical method of self-supervised learning [26], which has been broadly applied in mainstream computer vision tasks. It mainly focuses on identifying shared features across similar instances and differentiating among non-similar ones. MoCo [27] uses the idea of a dictionary to do contrastive learning by treating the outputs of two augmented pictures as query and key. SimCLR [28] reaches another milestone with the novel contrastive loss function, known as the normalized temperature-scaled cross-entropy loss (NT-Xnet), and projection head. Currently, a novel type of network, SimSiam [29], is gaining popularity since it proves that the simple Siamese network can still learn useful representations even in the absence of large batches, momentum encoder and negative pairs. In addition, it addresses issues with network collapse in collaboration with the stop-gradient operation, significantly improving the performance. Impressed by the creativity, we adopt the core design of SimSiam to elevate the representation capability of our encoder.

3 Methodology

In this section, we introduce the proposed automatic method in three parts: (1) detection branch: localizing all regions containing handwritten arithmetical exercises. (2) Recognition branch: parsing the cropped images and generating the corresponding exercise sequence. (3) Correction branch: designing templates to match the generated exercises for correction. The whole architecture is presented in Fig. 2. Specifically, the detection model first receives the input exercise image $\boldsymbol{M} \in \mathbb{R}^{3\times H\times W}$ and outputs three heat maps to localize arithmetical exercises. Then, each cropped arithmetical exercise $\boldsymbol{I^i}$ in the input image is inputted into our recognition model, obtaining the generated sequence $\{\boldsymbol{S}^i_1, \ldots, \boldsymbol{S}^i_T\}$ where $\boldsymbol{S}^i_j$ is the token of the arithmetical exercise in $\boldsymbol{I^i}$. Finally, we use some predefined templates to correct the arithmetical exercise.

3.1 Detection branch

This subsection explains our detection branch in detail, and the overall architecture is illustrated in Fig. 2. In order to excavate feature representation in multi-level, the ResNet-50 [20] is applied to extract feature maps $\{\boldsymbol{C_i} \mid i \in \{2,3,4\}\}$ at various scales at larger paces. Second, FPN [5] is used to fuse feature maps in multi-level, outputting fusion feature maps $\{\boldsymbol{P_i} \mid 3 \le i \le 7\}$. Then, the detection head locates arithmetical exercises in different scales. Moreover, "center-ness" [6] in the detection head is implemented to remove redundant or low-quality proposals.

3.1.1 FPN neck

Due to the structures of arithmetical exercises being multiple in scale, the size of proposals varies largely. Therefore, we introduce the FPN [5] mechanism to integrate multi-level features. And the detection head can locate multi-scale arithmetical exercises more accurately through the richer semantic information provided by FPN. We feed the arithmetical excise image into ResNet-50 [20] to get multi-level feature maps defined as $\{\boldsymbol{C_i} \mid i \in \{2,3,4\}\}$. Then, FPN fuses feature maps at adjacent scales, increasing the range of the receptive field. The specific process is shown below, where $\mathrm{Conv}(\cdot)$ denotes a series of convolution operations, and $\{\boldsymbol{P_i} \mid 3 \le i \le 7\}$ denotes the outputs of FPN. Due to the mechanism [5] of convolution neural networks, the receptive field increases with the number of convolution layers. That is, deep convolutional blocks tend to perceive large arithmetical exercises in our detection branch. Specifically, following the setting of FCOS [6], we apply the {Pi, $3 \le i \le 4$} to detect small mathematical exercises and $5 \le i \le 7$} to detect large ones, respectively.

$$\boldsymbol{P_i} = \begin{cases} \mathrm{Conv}(\mathrm{Conv}(\boldsymbol{C_i}) + \mathrm{Conv}(\boldsymbol{C_{i+1}})), & i = 3,4 \\ \mathrm{Conv}(\boldsymbol{C_i}), & i = 5 \\ \mathrm{Conv}(\boldsymbol{P_{i-1}}), & i = 6,7 \end{cases} \quad (1)$$

3.1.2 Detection head

The multi-scale detection heads are applied to locate arithmetical exercises at multi-level. These different detection heads share parameters, which enhances their

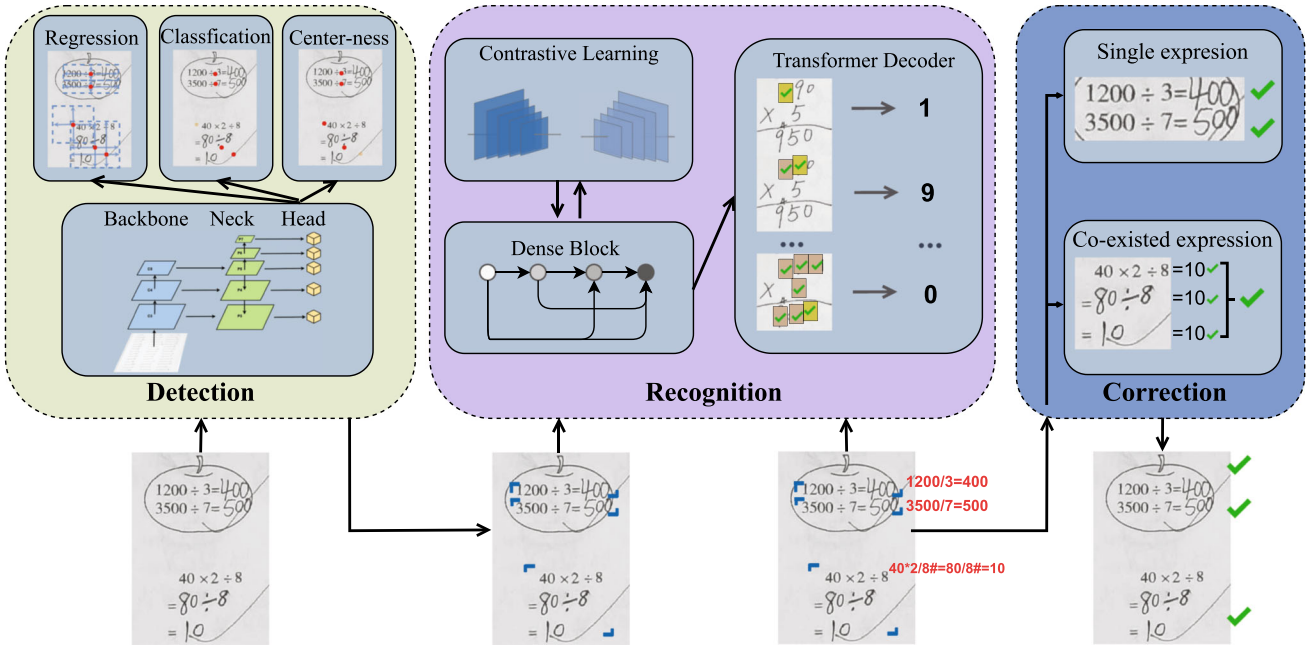

Fig. 2 The automatic method, FATE, is composed of three parts, i.e., detection branch, recognition branch and correction branch

representative capabilities and reduces the scale of the detection model. Each detection head consists of four stacked convolutional blocks. The following Eq. 2 defines the process, where $\{\boldsymbol{O}_i^j \mid 3 \le i \le 7, 0 \le j \le 4\}$ denotes the outputs of each block at different scales, and $\mathrm{GN}(\cdot)$ denotes group normalization. The final output $\boldsymbol{O}_i^4$, where i denotes different scales, includes the classification map, the regression map and the center-ness map.

$$\boldsymbol{O}_i^j = \begin{cases} \boldsymbol{P}_i, & j = 0 \\ \mathrm{ReLU}\left(\mathrm{GN}\left(\mathrm{Conv}\left(\boldsymbol{O}_i^{j-1}\right)\right)\right), & j = 1,2,3,4 \end{cases} \tag{2}$$

During the training phase, the ground truth of the i-th arithmetical exercise in the input image is considered as $g_i = \left(\tilde{x}_0^{(i)}, \tilde{y}_0^{(i)}, \tilde{x}_1^{(i)}, \tilde{y}_1^{(i)}, \tilde{c}^{(i)}\right)$, where the first four dimensions denote the coordinates of the top left corner and the bottom right corner in the proposal box, and the last dimension $\tilde{c}^{(i)}$ denotes the category of the proposal box. We take the location $(x^{(i)}, y^{(i)})$ of the pixel $o_{x,y}^{(i)}$ in the feature map $\boldsymbol{O}_i^4$ as the center coordinates of the proposal box. Then, let $o_{x,y}^{(i)}$ classifies its class $c_{x,y}^{(i)}$, calculates its center-ness $\delta_{x,y}^{(i)}$ and regresses its bias $\Delta_{x,y}^{(i)}$ from the neighboring ground truth to the location $(x^{(i)}, y^{(i)})$. For those pixels that fall into the overlapping region of multiple target boxes, we consider the target box with the smallest size as its label. In this paper, we define the ground truth of the bias as $\tilde{\Delta}_{x,y}^{(i)} = (\Delta\tilde{x}_0^{(i)}, \Delta\tilde{y}_0^{(i)}, \Delta\tilde{x}_1^{(i)}, \Delta\tilde{y}_1^{(i)})$.

$$\begin{aligned} \Delta\tilde{x}_0^{(i)} &= x_0^{(i)} - \tilde{x}_0^{(i)}, \Delta\tilde{x}_1^{(i)} = x_1^{(i)} - \tilde{x}_1^{(i)} \\ \Delta\tilde{y}_0^{(i)} &= y_0^{(i)} - \tilde{y}_0^{(i)}, \Delta\tilde{y}_1^{(i)} = y_1^{(i)} - \tilde{y}_1^{(i)} \end{aligned} \tag{3}$$

The detection model produces many redundant proposals without limiting the detection range of the multi-level detection heads, which lowers the detection accuracy. So, we add certain constraints to limit the regression range of the multi-level detection heads. When $\max(\tilde{\Delta}_{x,y}) \le k_i$ and $\max(\tilde{\Delta}_{x,y}) \ge k_{i-1}$, where $k_i \in \{0, 64, 128, 256, 512, \infty\}$, are satisfied, the pixel $o_{x,y}^{(i)}$ is regarded as a positive example, or it will be a negative example. This idea of partitioning allows multi-scale detection heads to perform different scale detection tasks, improving the overall detection.

3.1.3 Center-ness

Arithmetical exercises can be divided into single and co-existed equations. Specifically, a single equation only contains one simple equation, while a co-existed equation usually contains many simple equations that depend on each other. As shown in Fig. 2, the structure of the multiple single equations is similar to the structure of one single co-existed equation, which leads to ambiguity in detection boundaries. This misleads the model to produce redundant low-quality proposals. To solve this problem, the geometric centers of different equations are considered to distinguish the difference between these two exercises. Concretely, we add the center constraints, dubbed as "center-ness," which represents the normalized distance

from the location to the center of the arithmetical exercise. The center-ness $\delta_{x,y}$ [6] is calculated on the classification branch as illustrated in Fig. 2 and suppresses the redundant proposals by reducing the confidence of proposals predicted by non-central pixels. During the training phase, its label $\tilde{\delta}_{x,y}$ can be denoted by the following Eq. 4.

$$\tilde{\delta}_{x,y} = \sqrt{\frac{\min(\Delta\tilde{x_0}, \Delta\tilde{x_1})}{\max(\Delta\tilde{x_0}, \Delta\tilde{x_1})} \times \frac{\min(\Delta\tilde{y_0}.\Delta\tilde{y_1})}{\max(\Delta\tilde{y_0}, \Delta\tilde{y_1})}} \tag{4}$$

Finally, we optimize the loss function $\mathcal{L}(o_{x,y}, \Delta_{x,y})$ defined in Eq. 5, where $\mathcal{L}_{\text{cls}}$ denotes the focal loss [30], $\mathcal{L}_{\text{reg}}$ denotes the IOU loss in UniBox [28] and $\mathcal{L}_{\text{bce}}$ denotes the cross-entropy of the binary classification. $\mathbb{1}_{\{\tilde{c}_{x,y}>0\}}$ denotes the indicator function, being 1 if $\tilde{c}_{x,y} > 0$ and 0 otherwise.

$$\begin{aligned}\mathcal{L}(o_{x,y}, \Delta_{x,y}) = &\frac{1}{N_{\text{pos}}} \sum_{x,y} \mathcal{L}_{\text{cls}}(c_{x,y}, \tilde{c}_{x,y}) + \mathcal{L}_{\text{bce}}(\delta_{x,y}, \tilde{\delta}_{x,y}) \\ &+ \frac{\lambda}{N_{\text{pos}}} \sum_{x,y} \mathbb{1}_{\{\tilde{c}_{x,y}>0\}} \mathcal{L}_{\text{reg}}\left(\Delta_{x,y}, \tilde{\Delta}_{x,y}\right)\end{aligned} \tag{5}$$

In the testing process, the confidence of each proposal is calculated by multiplying the center-ness score and the classification score. Those proposals with confidence higher than the predefined confidence threshold are reserved. As illustrated in Fig. 2, we suppress those proposals with low confidence scores whose center pixel is colored orange.

3.2 Recognition branch

In the recognition branch, the encoder–decoder framework is adopted as the overall architecture to accomplish the image-to-text task. First, given an image containing one arithmetical exercise cropped from the detection branch, the encoder extracts high-level semantic information from it to generate the feature map $\boldsymbol{F}$. Then, the image positional encoding matrix $\boldsymbol{PE}^{I}_{x,y,d}$ is added to it for enhancing the time-series attribute in (x,y) dimension. Finally, the feature map with positional information is fed into the decoder for obtaining the arithmetical exercise $\{\boldsymbol{S}_1, \ldots, \boldsymbol{S}_T\}$.

We apply the DenseNet [31] to derive the feature map $\boldsymbol{F}$ of the input cropped arithmetical exercises due to its strong ability of image feature extraction. Simultaneously, the self-supervised contrastive learning method is introduced to prompt the representation capability of the encoder for better performance. The transformer-based decoder module, which combines a bidirectional training strategy for capturing the global semantic dependencies, is mainly discussed in Sect. 3.2.2. The entire view of the model is illustrated in Fig. 3a.

3.2.1 Encoder

We employ the densely connected convolutional network (DenseNet) [31] as the encoder to extract the feature of the cropped arithmetical image. With a series of dense blocks and dense connections between feature maps, the encoder maintains the completeness of complex spatial structural features of arithmetical exercises. Assume that $\boldsymbol{F_l}$ is the feature map output of the $(l-1)$th dense block, then $\boldsymbol{F_{l+1}}$ can be computed by:

$$\boldsymbol{F_{l+1}} = \mathrm{T_l}(\mathrm{D_l}([\boldsymbol{F_0}; \boldsymbol{F_1}; \ldots; \boldsymbol{F_l}])) \tag{6}$$

where $\mathrm{D_l}(\cdot)$ denotes the dense block operation which is the composite of three layers, i.e., 3×3 convolutional layer, BN and activation layer. $\mathrm{T_l}(\cdot)$ denotes the transition block between two dense blocks, which consists of a 1×1 convolutional layer and a 2×2 average pooling layer.

In a nutshell, DenseNet inputs the cropped exercise image $\boldsymbol{I^i} \in \mathbb{R}^{3\times H'\times W'}$ by the detector and outputs the feature map with C channels. Considering that handwritten arithmetical exercises are mostly in a slender format, although the shapes of images in the dataset are various, resizing all images to 64×256 is feasible based on the experiment.

After obtaining the feature map, image positional encoding can enrich time-series information for the decoder. The operation is similar to word positional encoding [23] with the difference that the input of image positional encoding is two-dimensional. Particularly, sinusoidal positional encoding vector $\boldsymbol{PE}^{W}_{\bar{x},d}$ and $\boldsymbol{PE}^{W}_{\bar{y},d}$ are computed in the x and y dimensions, respectively. They are then concatenated for the image positional encoding matrix. Given a 2D position tuple (x; y) and the dimension d as the word positional encoding, the image encoding vector $\boldsymbol{PE}^{I}_{x,y,d}$ can be represented as follows:

$$\begin{aligned}\bar{x} = \frac{x}{H}, \quad \bar{y} = \frac{y}{W} \\ \boldsymbol{PE}^{I}_{x,y,d} = \left[\boldsymbol{PE}^{W}_{\bar{x},d/2}; \boldsymbol{PE}^{W}_{\bar{y},d/2}\right]\end{aligned} \tag{7}$$

3.2.2 Decoder

We adopt the standard transformer [23] with essential parts as the decoder. First, the flattened feature map with positional information from the encoder is input to the decoder as the key and query, which are employed to compute the multi-head scaled attention with the value matrix, i.e., the symbol label of the arithmetical exercise during training, or

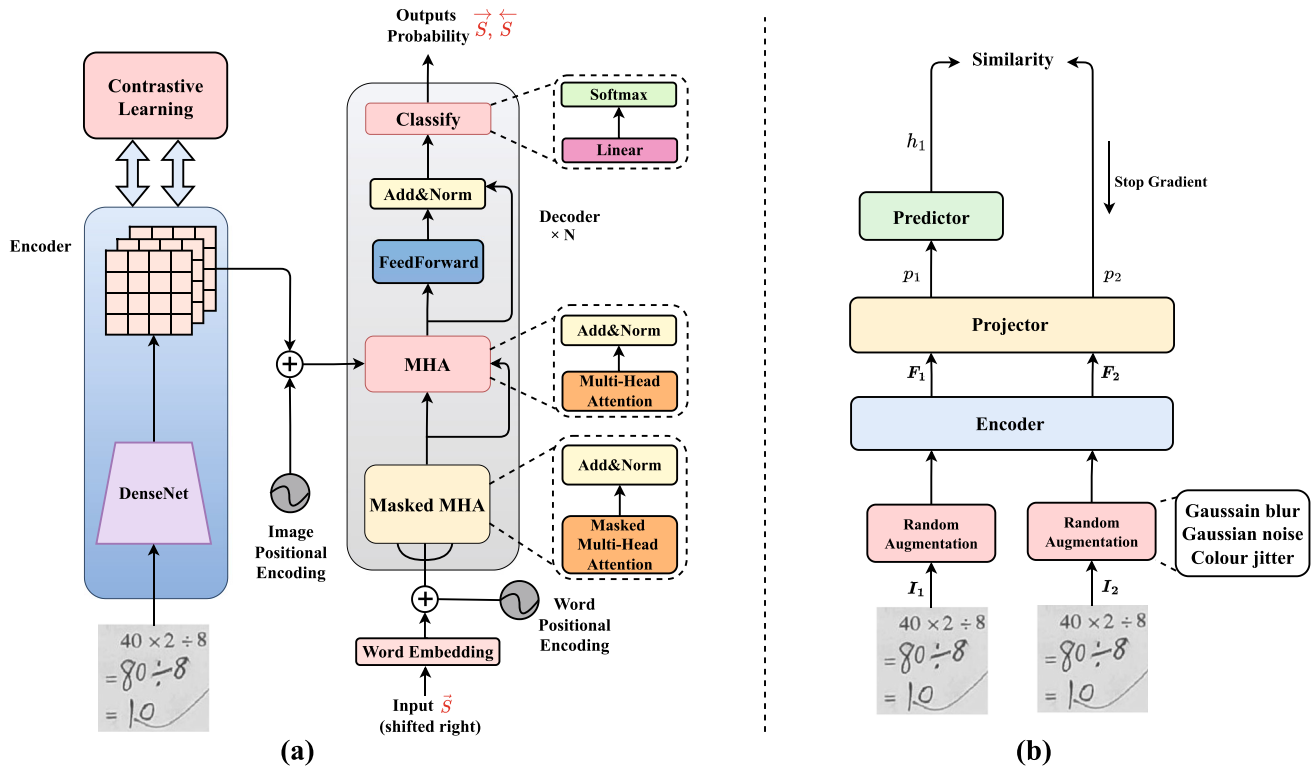

Fig. 3 Illustration of the overall architecture of the recognition branch. The sub-figure (**a**) describes components of the recognition branch, and sub-figure (**b**) elaborately demonstrates the design of the contrastive learning module

output sequence of the decoder in preceding steps during testing. This operation excavates the similarity between the symbol sequence and the global image information, enhancing the perception for multi-scale arithmetical exercises. Specifically, assuming that $\boldsymbol{K}$, $\boldsymbol{Q}$ and $\boldsymbol{V}$ denote key, query and value matrix, respectively, the multi-head attention can be illustrated as follows:

$$\begin{aligned} \text{MultiHead}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) &= \text{Concat}(\text{head}_1,\ldots\text{head}_\text{h})\boldsymbol{\Theta} \\ \text{head}_\text{i} &= \text{Attention}\left(\boldsymbol{Q}\boldsymbol{\Theta}_i^{\boldsymbol{Q}},\boldsymbol{K}\boldsymbol{\Theta}_i^{\boldsymbol{K}},\boldsymbol{V}\boldsymbol{\Theta}_i^{\boldsymbol{V}}\right) \end{aligned} \tag{8}$$

where $\text{Attention}(\cdot)$ stands for the scaled dot product attention operation. $\boldsymbol{\Theta}_i^{\boldsymbol{Q}} \in R^{d\times d_k}$, $\boldsymbol{\Theta}_i^{\boldsymbol{k}} \in R^{d\times d_k}$, $\boldsymbol{\Theta}_i^{\boldsymbol{k}} \in R^{d\times d_k}$, $\boldsymbol{\Theta} \in R^{d\times d}$,$d_k = d\text{mod}(\text{head})$ and by default, the number of heads is set to 8. The attention score matrix which measures the correspondence between the feature map and the label is reformatted into the feed-forward Layer and linear layer with softmax function to generate the arithmetical sequence.

Referring to the general approach [23], we adopt the auto-regressive method in both training and inference. Notably, the bi-directional decoding strategy [3] is added to improve the recognition accuracy in the training phase. For the target arithmetical exercise sequence $\boldsymbol{S}_{\text{target}} = \{s_1,\ldots,s_T\}$, the corresponding directional sequences can be represented as follows:

$$\begin{aligned} \overrightarrow{S} &= \{\langle\text{SOS}\rangle, \text{s}_1,\ldots,\text{s}_\text{T},\langle\text{EOS}\rangle\} \\ \overleftarrow{S} &= \{\langle\text{EOS}\rangle, \text{s}_\text{T},\ldots,\text{s}_1,\langle\text{SOS}\rangle\} \end{aligned} \tag{9}$$

where T is the length of the sequence, and $\langle\text{SOS}\rangle$ and $\langle\text{EOS}\rangle$ are special tokens. Assuming that $\overrightarrow{S}^*$ is the sequence generated from left to right, and $\overleftarrow{S}^*$ is generated from right to left, the generation process is given by:

$$\overrightarrow{S}^* = \underset{s}{\text{argmax}} \sum_{i=0}^{\text{T}-1} \log p(s_{i+1} \mid s_{\le i}, v, \sigma) \tag{10}$$

where $s_{\le i} = \{s_1,\ldots,s_i\}$, v is the feature vector generated by the encoder from feature map $\boldsymbol{F}$, and σ is the parameter set of the transformer decoder. The generation process of $\overleftarrow{S}^*$ is similar. Cross-entropy function is chosen to compute the loss in both directions, which can be expressed as follows:

$$\mathcal{L}_{ce}(\theta) = -\sum_{i=1}^{T} \log p(\vec{s}_{i+1} \mid \vec{s}_{\le i}, \sigma). \tag{11}$$

3.2.3 Contrastive learning paradigm

Arithmetical exercises can be divided into a number of categories, and those within the same category share

similar features. In order to extract the distinctive feature of each type of arithmetical exercises, which can further strengthen its representational capability, a contrastive learning paradigm is employed in the training phase. As illustrated in Fig. 3b, the additional projector and predictor are adopted based on the Simsiam [29] manner. Specifically, the encoder simultaneously takes two randomly augmented views $\boldsymbol{I_1}$ and $\boldsymbol{I_2}$ from the cropped exercises image $\boldsymbol{I}$ as input and outputs corresponding feature maps $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$, respectively. Notably, the random augmentation operation contains Gaussian blur, color jitter and added Gaussian noise. Then, the projector module takes the two flattened $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$ as input and outputs intermediate features p_1 and p_2. Finally, the predictor contains two MLP blocks to generate the output symbol. It maps p_1 and p_2 to final contrastive features h_1 and h_2. To learn the relationship between semantic information and representation space, the cosine similarity function is applied to measure the similarity, given by:

$$\mathcal{D}(p_1, h_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{h_2}{\|h_2\|_2}. \tag{12}$$

In light of the BYOL [32], the symmetric loss is defined as follows:

$$\mathcal{L}_{\text{con}} = \frac{1}{2}\mathcal{D}(p_1, h_2) + \frac{1}{2}\mathcal{D}(p_2, h_1). \tag{13}$$

Significantly, for the purpose of preventing collapsing of the encoder, the gradient of h_2 is stopped to propagate back to the encoder, which serves as the constant. Conversely, the encoder receives the gradient from p_2. The process can be illustrated as follows:

$$\mathcal{L}_{\text{con}} = \frac{1}{2}\mathcal{D}(p_1, \text{stopgrad}(h_2)) + \frac{1}{2}\mathcal{D}(p_2, \text{stopgrad}(h_1)) \tag{14}$$

where $\mathcal{L}_{\text{con}} \in [-1, 1]$. Noteworthy, the simple combination between $\mathcal{L}_{\text{ce}}$ and $\mathcal{L}_{\text{con}}$ causes unexpected gradient values when conducting backpropagation. As a consequence, the value of the contrastive loss function is necessary to remain constantly positive. Furthermore, for balancing weights of two different losses, the coefficient $\alpha = 0.5$ scales the range of contrastive loss between 0 and 1. The loss is adjusted as:

$$\mathcal{L}_{\text{mod}} = \alpha \cdot \max(0, 1 - \mathcal{L}_{\text{con}}). \tag{15}$$

Therefore, the total loss of the recognition branch is as follows:

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{ce}} + \beta \cdot \mathcal{L}_{\text{mod}}, \tag{16}$$

where β is set to 1 by default.

3.3 Correction branch

In order to achieve automatic correction, the union of nine templates with corresponding matching rules and correction criteria is predefined as the "prefabricated template library." As shown in Fig. 4, the entire procedure of the correction branch can be summarized as three steps. First, the pre-processing module processes the arithmetical sequence that is produced by the recognition branch. Specifically, two key procedures are included: (1) extract $=$, $<$ and $>$ wrapped in parentheses and (2) transform all kinds of parentheses into round parentheses. Notably, the pre-correction module would directly judge the exercise as wrong if the incomplete arithmetical sequence where the answer is missing or the parentheses are not in closure. Second, the arithmetical sequence is classified as an exclusive template from the library of prefabricated templates. For instance, the recognized arithmetical sequence from the recognition branch, "$84/2 + 3\# = 42 + 3\# = 45$," is adjusted in format and prior corrected through pre-process pre-correction. Then, it is identified as the recursive equation by the unique token "#=." Finally, corresponding to the recursive expression template, the generated sequence is examined with the consistency and accuracy of intermediate expressions. The judgment is correct only if all intermediate expressions are equal to the final result. Figure 5 shows different correction results based on the branch to demonstrate the superior performance of our model.

4 Experiments

4.1 Dataset and evaluation

Dataset: We use the handwritten arithmetical exercise dataset AEC-5k [1] to evaluate our proposed method, which contains various forms of arithmetical exercises. In detail, this dataset consists of 5000 mathematical homework images of elementary school students and 77,097 arithmetical exercises. We split AEC-5k into the training set and the test set with a ratio of 9:1, which are dubbed AEC-4.5k and AEC-500, respectively. Since there are various structures and forms of arithmetical exercise, we classify them from two perspectives (shown in Fig. 6), i.e., equation type and equation format. Specifically, an equation is deemed "simple" if it conforms to the following criteria: (1) It is presented in a single line; and (2) it lacks any units, fractions or remainders. In contrast, the "recursive equation" based on its multi-line format and multiple instances of the "=" sign is co-existed equations. While the

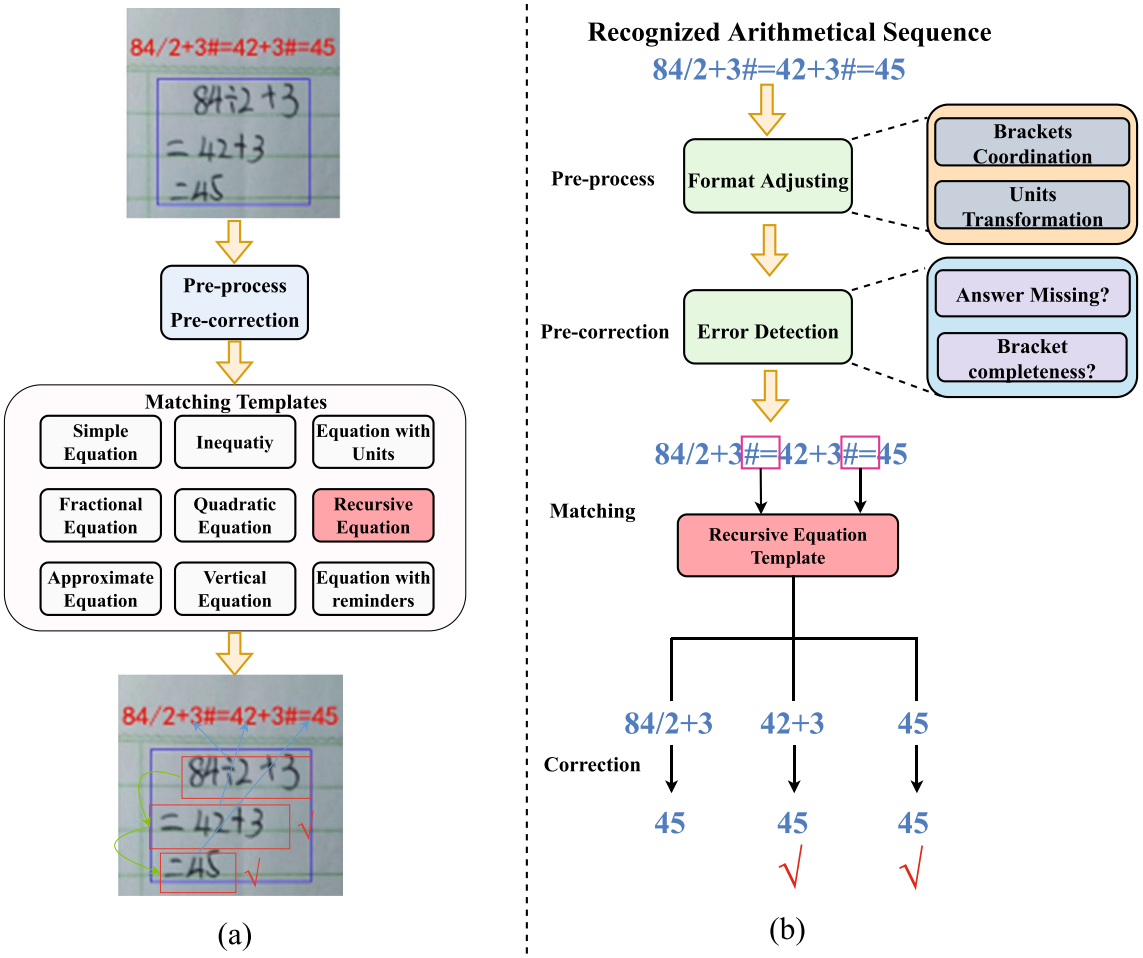

Fig. 4 Illustration of the entire process of correction branch. Note that the text in red denotes the recognized arithmetical sequence that represents the arithmetical exercise in the image. The content in the red box is the intermediate component of the equation in sub-figure (**a**). The sub-figure (**b**) clearly demonstrates the entire correction process

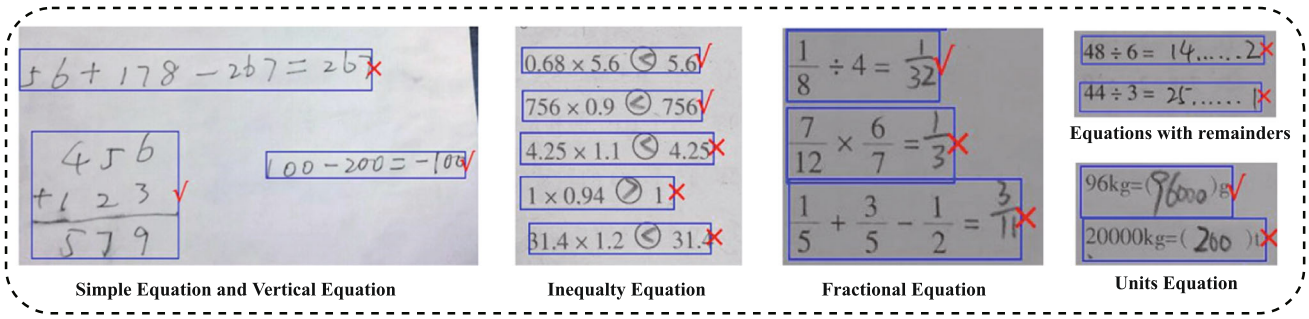

Fig. 5 Comparison of different correction results of arithmetical exercises

equation type is intended for the subsequent correction branch. Meanwhile, the classification method based on the format tries to describe the structure of exercises, e.g., unit equations, fraction equations and vertical equations.

Evaluation: For the detection branch, we select the evaluation criterion following FCOS [6]. First, we calculate the average precision and average recall to evaluate the effectiveness of the detection branch. Second, we investigate the AP and AR for exercises of different scales. "L", "M" and "S" denote the exercise for the three sizes: large, medium and small. Then, we choose AP_L, AP_M, AP_S, AR_L, AR_M and AR_S to denote the average precision and average recall of the exercises of different scales.

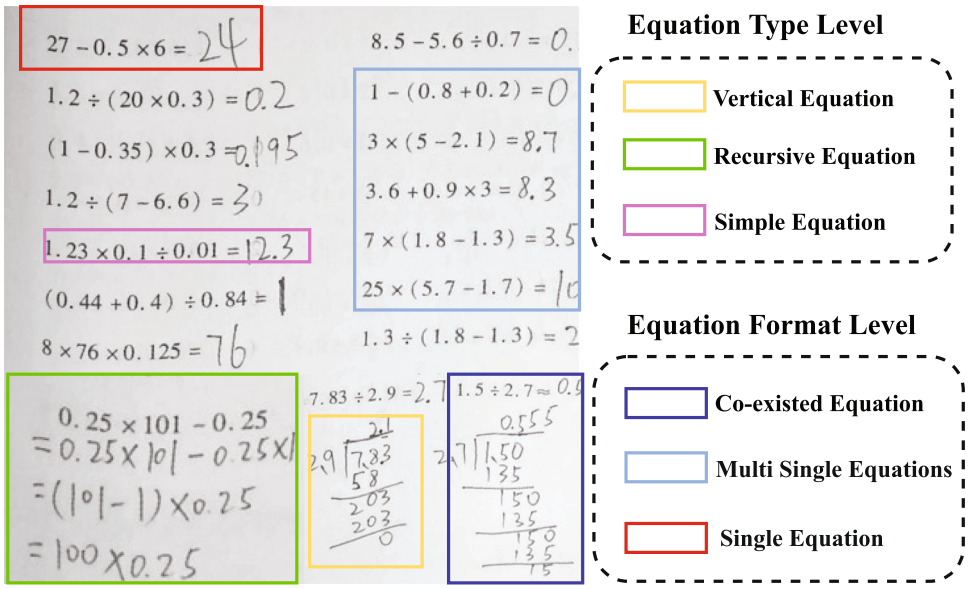

Fig. 6 Illustration of different classifications of equations. The type level is based on equation type for the purpose of corrections. The format level is based on the equation format for detection

To evaluate the effectiveness of the recognition branch convincingly, we directly crop images according to the ground truth and regard them as the input of the recognition model. We select three quantitative metrics of accuracy, i.e., Exprate, ≤ 1 and ≤ 2, which indicate the exercise recognition rate when zero to two structural or symbol errors can be tolerated. Meanwhile, we conduct the spotting experiment to verify the overall performance of the model. In detail, we first locate the arithmetical exercise by the detection model and crop it based on the location. Then, we feed these cropped images into the recognition model to obtain the final spotting results. We utilize the precision (P), recall (R) and F_1 to measure the spotting performance of different combinations of prevalent detection and recognition models, given by:

$$P = \frac{N_{rec-right}}{N_{true}}, R = \frac{N_{rec-right}}{N_{det}}, \quad F_1 = \frac{2 \times P \times R}{P + R}, \tag{17}$$

where $N_{rec-right}$ denotes the number of correctly recognized arithmetical exercises, N_{true} denotes the true number of arithmetical exercises and N_{det} denotes the number of detected arithmetical exercises.

4.2 Experiment implementation

Training details: To avoid the accumulation of errors, we train the detection branch and the recognition branch, respectively. For detection, we finetune our detection branch with a TITAN RTX GPU based on the AEC-4.5K dataset and the pre-trained model [6], which is trained on the large-scale detection benchmark COCO. The SGD [33] optimization scheme with an initial learning rate of 0.005 and momentum of 0.9 is used to optimize the network. A common learning rate adjustment strategy is employed to realize the learning rate decay, that is, the learning rate is decayed by a factor of 0.1 when 60,000 iterations and 80,000 iterations. For recognition, we train the recognition branch with 4 × RTX 3090 GPUs based on about 70 thousand arithmetical exercises from the AEC-4.5k. The AdaDelta [34] optimization scheme with an initial learning rate of 1.0 and weight decay of 1e−4 is used to optimize the network. For data augmentation, Gaussian blur, random Gaussian noise and color jittering are adopted for recognition accuracy improvements.

Inference details: We evaluate our model on AEC-500. The product of the center-ness score and the class score serves as the confidence of proposals. We set the non-max suppression threshold to 0.7 to suppress those low-quality proposals. After obtaining cropped images from the detection branch, we feed them into the recognition branch. Specifically, we use beam search with ten beams when inference.

4.3 FATE: ablation study

In this section, we conduct thorough ablation studies to verify the effectiveness of the detection and recognition branches, respectively. All studies in this section are conducted with AEC-5k.

4.3.1 Detection branch

Multi-level prediction with FPN: In this study, we evaluate the benefit of FPN. Of note, we investigate the exercise dataset and find that most of the exercises are small in size. At the same time, the variance of the exercise size distribution is large. Therefore, we compare the effectiveness of

multi-level detection between the model with FPN and without FPN. In Table 1, it can be noticed that AP is improved by 0.6% with the addition of FPN network branches. In particular, AR and AR_S are improved by 1.2% and 0.5%, respectively.

We also add cost-free improvements in our detection branch as shown in Table 1. These tricks successfully boost our detection branch from 96.0 to 96.8% in AP. In addition, the center sampling strategy makes our detection branch accurately locate small exercises and improves AP_S and AR_S by 0.9% and 0.4%.

Center-ness: The following study verifies the effectiveness of center-ness. The method of the third line in Table 2 ("center-ness on reg") is the full version of our method with all improvements in Table 1. In addition, the methods corresponding to the other two lines are the variants of the method corresponding to the third line. Typically, the ground truth of the boundary of exercise is ambiguous, which results in the occurrence of redundant bounding boxes. To justify whether the center-ness scheme addresses the above problem, we perform the comparison experiment of center-ness based on multi-scale arithmetical exercises. Table 2 reveals our detection model with center-ness branch boosts AP_S from 93.9 to 98.1% and AR_S from 95.4 to 97.0%.

4.3.2 Recognition branch

In this study, we conduct experiments to verify the effectiveness of per component. For efficiency, the recognition model we use is the FATE recognition branch with four dense blocks.

Image positional encoding: In this study, we evaluate the benefit of image positional encoding. Statistically, arithmetical exercises mostly appear orderly in an image. To capture this property, image positional encoding is thought to be a reasonable way. Therefore, Table 3 demonstrates that image positional encoding makes a difference in the final result. It verifies our assumptions that the image positional encoding technique enriches the time-series information of the feature map. Therefore, the Exprate metric is increased by 5.9%.

Contrastive learning paradigm: In the proposed recognition branch shown in Fig. 3, we adopt the contrastive learning paradigms. Specifically, the data augmentations included with added Gaussian noise, Gaussian blurring and color dithering are used in our implementation. It not only creates two different views of the same image but also amplifies the diversity of data. The results are reported in Table 3, and we discover that the model gains further improvement in Exprate on top of the already high recognition accuracy rate. This, in turn, demonstrates the effectiveness of the end-to-end contrastive learning method, which promote the generalization ability of the encoder to extract more high-level hidden semantic information from an exercise image.

4.4 FATE: comparison with state-of-the-art

We compare our FATE with several widely-used detection and recognition models, respectively. Furthermore, we conduct end-to-end spotting experiments to demonstrate the capability of FATE for arithmetical exercise correction. All the compared methods and variants follow the same training strategy as our model. Notably, we replace the ResNet-50 [20] with ResNet-18 [20] as the encoder of

Table 1 Comparison of the impact of FPN neck and some cost-free improvements on our network

Method	w/o FPN[a]	AP	AP^b_S	AP^b_M	AP^b_L	AR	AR_S	AR_M	AR_L
FATE detection	✗[c]	95.4	93.7	95.3	95.5	97.5	96.1	97.5	97.4
FATE detection	✔[c]	96.0	94.9	96.0	96.1	97.8	96.6	97.9	97.7
Improvements									
+ GN. in head[d]	✔	96.5	95.5	96.6	96.5	98.2	97.0	98.4	98.1
+ ctr. on reg[e]	✔	96.7	94.2	**96.8**	96.9	**98.4**	96.6	**98.5**	98.3
+ ctr. sampling[f]	✔	**96.8**	**95.1**	96.8	96.9	98.4	**97.0**	98.5	**98.4**

[a] "w/o FPN" indicates whether to add FPN network branches to the original network

[b] "L," "M" and "S": the exercise for the three sizes of large, medium and small, respectively

[c] "✔" indicates that the model is implemented with the corresponding technique, while "✗" is on the contrary

[d] "GN. in head": group normalization in the head

[e] "ctr. on reg": computes center-ness by with the regression branch and by default in the classification branch center-ness

[f] "ctr. sampling": perform center-ness sampling

Table 2 Comparison of the impact of center-ness branches on the network, respectively

Method	AP	AP_S	AP_M	AP_L	AR	AR_S	AR_M	AR_L
None[a]	96.3	93.9	96.1	96.5	98.0	95.4	98.0	98.1
Center-ness on cls.[b]	96.5	94.4	96.5	96.5	98.2	96.3	98.4	98.2
Center-ness on reg[c]	**96.8**	**95.1**	**96.8**	**96.9**	**98.4**	**97.0**	**98.5**	**98.4**

[a]"None": the center-ness branch is not used

[b]"center-ness on cls.": center-ness is computed on the classification branch

[c]"center-ness on reg.": center-ness is computed on the regression branch

Table 3 Comparison of the impact of "IPE" and "CL" on the network, respectively

Model	IPE[a]	CL[b]	ExpRate	≤ 1	≤ 2
FATE	✗	✗	85.5	92.8	95.6
	✔	✗	91.4	95.7	97.4
	✔	✔	**91.7**	**95.9**	**97.6**

[a]"IPE": image positional encoding

[b]"CL": contrastive learning

ANMT in AEC [1], since the ANMT with ResNet-50 cannot maintain stable training process in AEC-4.5k.

4.4.1 Detection branch

As Table 4 shows, our detection model outperforms these mainstream detection models and the tailored detection method ScanSSD [8] significantly. The reasonable structure and anchor-free detection method make our model locate exercises more accurately than others. In addition, our detection is simple yet performs excellent performance in detecting small exercises. Specifically, our best model achieves 96.8% in AP and 98.4% in AR.

4.4.2 Recognition branch

To demonstrate the superiority and adaptability of our method in arithmetical exercises recognition, we compare ours with the HMER model CAN [2] and ABM [4] trained on AEC-4.5k and the recognition branch of AEC [1]. To ensure the fairness of performance comparison, we use the same data augmentation operations. As shown in Table 5, our model outperforms the state-of-the-art arithmetical exercise correction method, i.e., AEC, of 26.5% in Exprate. Compared with the state-of-the-art HMER methods, our approach delivers significant improvements in all metrics.

To verify the validity of the FATE recognition branch, we further implement comparison experiments between different variants of our model. Table 5 convinces us that the encoder plays an important role in recognition accuracy. The recognition performance of the FATE increases significantly with the number of dense blocks. Therefore,

Table 4 Comparison experiment of the detection branch

Method	BackBone	AP	AP_S	AP_M	AP_L	AR	AR_S	AR_M	AR_L
Two-stage methods									
Faster-RCNN [15] w/ FPN [5][a]	ResNet-50	74.6	65.7	73.4	75.5	80.1	71.4	79.1	81.2
One-stage methods									
ScanSSD [8]	Vgg-16	66.7	57.1	64.8	68.4	74.1	65.2	73.0	75.3
YOLOv3 [16]	Darknet-53	68.5	59.8	67.0	69.8	75.2	67.1	73.5	76.7
YOLOf [18]	ResNet-50	76.1	68.2	75.1	76.9	81.1	73.2	80.4	81.8
RetiNaNet [30]	ResNet-50	75.1	70.3	74.3	75.9	81.0	75.0	80.4	81.6
DETR [17]	ResNet-50	73.6	48.9	72.1	75.2	79.9	64.5	78.3	81.4
CenterNet [10]	Hourglass-52	68.2	57.9	73.6	63.7	75.6	68.0	80.1	71.8
FATE detection	ResNet-50	96.0	94.9	96.0	96.1	97.8	96.6	97.9	97.7
FATE detection[b†]	**ResNet-50**	**96.8**	**95.1**	**96.8**	**96.9**	**98.4**	**97.0**	**98.5**	**98.4**

[a] Faster-RCNN w/ FPN: faster-RCNN network [15] with FPN network [5]

[b] FATE Detection†: FATE detection branch with improvements

Table 5 Comparison experiment of the recognition branch

Method	Encoder	Decoder	Exprate	≤ 1	≤ 2
ANMT [1]	ResNet-18 + MDLSTM	LSTM	65.8	81.3	82.8
ABM [4]	DenseNet	MutalGRU	91.5	95.9	97.5
CAN [2]	DenseNet	Countingdecoder + GRU	82.8	91.1	94.1
FATE	DenseNet-B3[a]	Transformer	88.5	93.7	95.8
	DenseNet-B4	Transformer	91.7	95.9	97.6
	DenseNet-B5	Transformer	**92.3**	**96.3**	**97.7**

[a]"B3": three dense blocks

we adopt the DenseNet-B5, which performs superior performance in three metrics, as the encoder of the recognition branch of our final FATE model.

4.4.3 Spotting

To evaluate the effectiveness of our end-to-end method, we carry out the spotting experiment to compare different combinations of detection and recognition models. That is, we use the proposals generated by different detection models as the input to the recognition models. We compare our FATE with the AEC [1] and some well-known variants of detection and recognition branch, e.g., YOLOf [18], CAN [2] and ABM [4]. The performance of different approaches is shown in Table 6. In light of the results, the AEC performs poorly on AEC-500 (see in the first row), while the FATE achieves the best performance in the spotting experiments. Notably, our recognition branch improves the spotting performance even if the detection branch is changed to another model, showing the generalization of our design. Furthermore, we conduct qualitative studies by comparing the visualization of detection and recognition results between different variants. The results shown in Fig. 7 reveal that (1) our FATE can tackle the complicated boundaries agilely and prevent producing redundant bounding boxes; (2) the compared methods struggle with the problems of overlap and similar symbols, while our method generates sequences consistent with the content of the image and (3) although manually introducing Gaussian noise to the original image or encountering the situation where the exercises are tilted, our model can tackle them in an effective way. The stunning results again demonstrate the superior performance of our proposed method.

4.4.4 Failure cases

Arithmetical exercise correction is an intricate task due to its uncertainty and complexity. To further analyze the challenge of this task, typical error cases are presented in Fig. 8, which highlights three primary causes associated with incorrect results. (1) Low image quality, e.g., heavy background noise, low image resolution and lightness deviation. These factors interfere with the model to learn key feature information. (2) Equations with complex structures pose a challenge for the model to localize and identify the boundaries and structures of equations. (3) Similar symbols, which confuse the models and lead to redundant detection and wrong recognition results. We believe that these analyses can provide inspiration for future direction.

Table 6 Comparison experiment of spotting

Method		Spotting		
Detection	Recognition	P	R	F_1
AEC detection [1]	ANMT [1]	63.6	57.0	60.1
YOLOf [18]	ABM [4]	55.6	88.5	68.3
YOLOf [18]	CAN [2]	49.3	81.4	61.4
YOLOf [18]	FATE recognition	56.6	89.3	69.3
FATE detection	ABM [4]	90.8	82.4	86.4
FATE detection	CAN [2]	82.2	90.1	86.0
FATE detection	FATE recognition	**91.7**	**90.8**	**91.2**

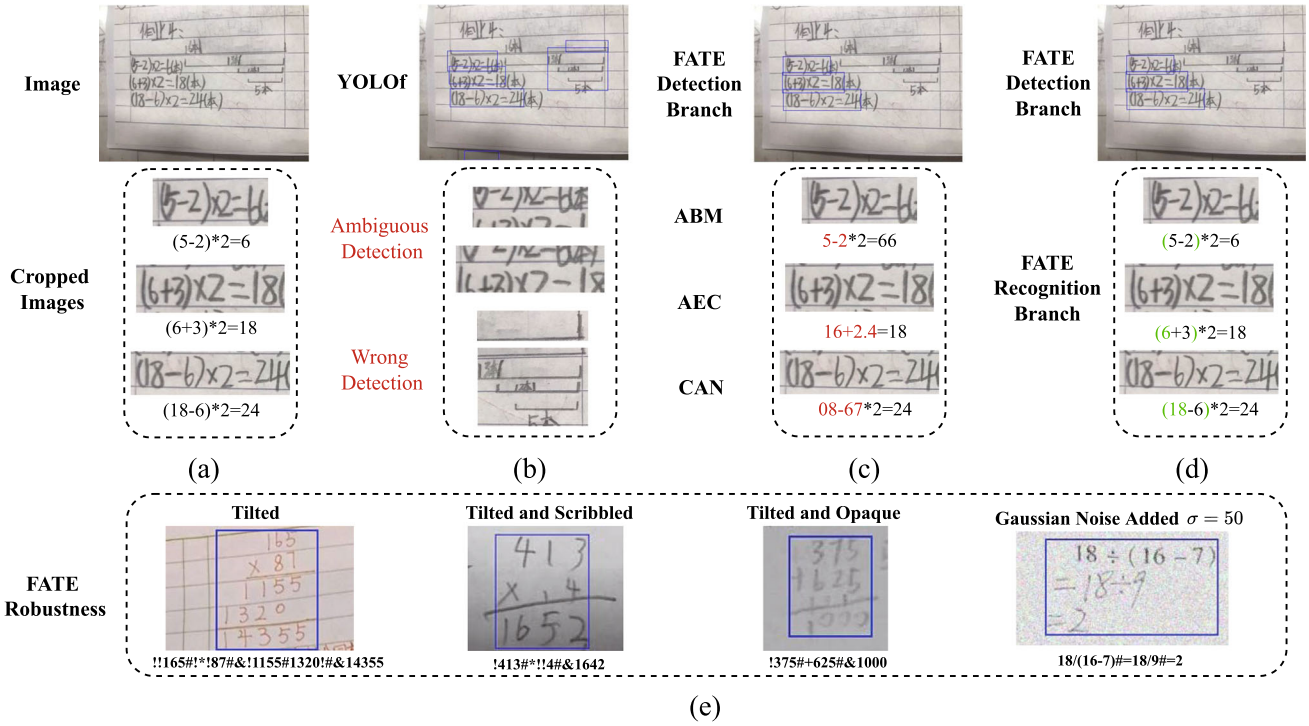

Fig. 7 Visualization of the results of different combinations of detection and recognition models. The sub-figure (**a**) is the input image and the ground truth cropped images with arithmetical exercises. The sub-figure (**e**) shows the robustness of FATE. Note that the red denotes errors while the green indicates correctness

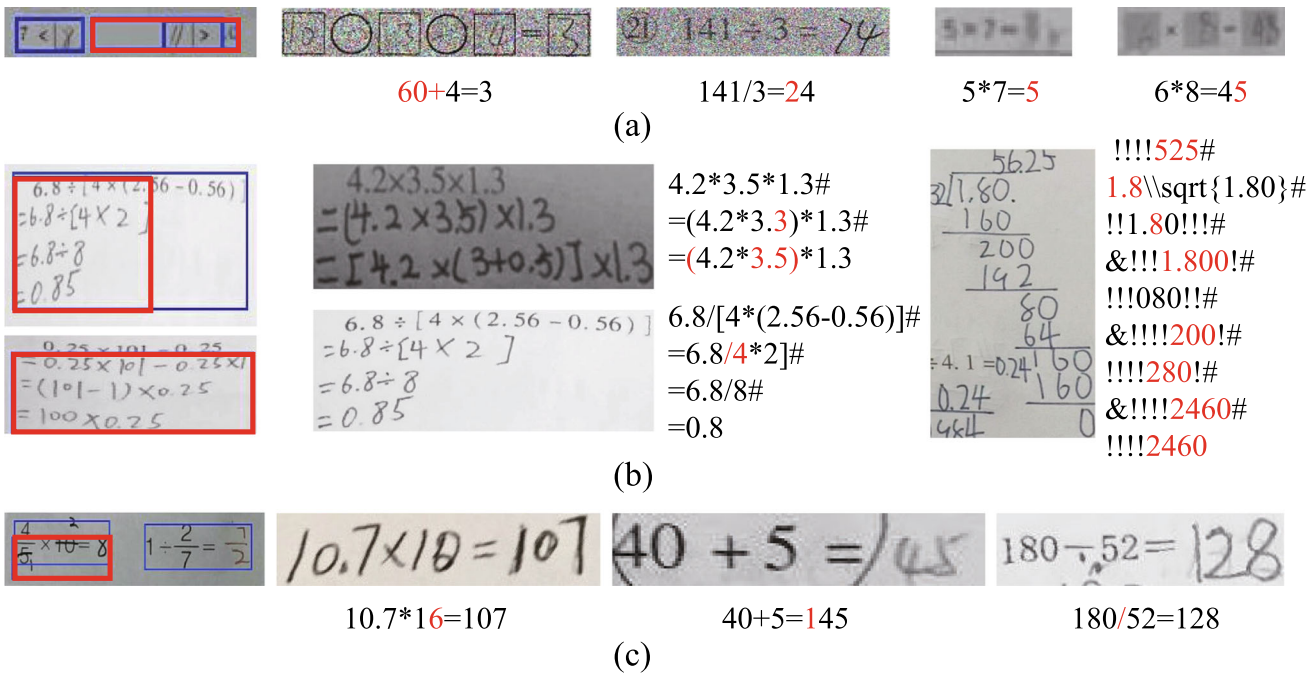

Fig. 8 The failure cases of the detection (first row) and recognition branches (other rows), where (**a**), (**b**) and (**c**) correspond to the three factors of low image quality, i.e., complex structure and distorted text, respectively. The red rectangle in detection cases denotes the wrong localization bounding box, while the red symbols in recognition cases denote the wrong counterparts

5 Conclusion

In this paper, we present a novel end-to-end three-stage method, FATE, which can auto-correct arithmetical exercises for elementary school students. We apply the anchor-free detection model, encoder–decoder framework and algorithm templates to locate, recognize and correct exercises. Furthermore, comprehensive experiments and ablation studies on AEC-5k demonstrate that our model outperforms other models and shows promising results.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 62076062) and the Social Development Science and Technology Project of Jiangsu Province (No. BE2022811). Furthermore, the work was also supported by the Collaborative Innovation Center of Wireless Communications Technology and the Big Data Computing Center of Southeast University.

Data Availability The AEC-5k dataset analyzed during the current study is available in the TencentYoutuResearch repository, with the link: https://github.com/TencentYoutuResearch/OCR-AEC5k.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Hu Y, Zheng Y, Liu H, Jiang D, Liu Y, Ren B (2020) Accurate structured-text spotting for arithmetical exercise correction. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 686–693
2. Li B, Yuan Y, Liang D, Liu X, Ji Z, Bai J, Liu W, Bai X (2022) When counting meets hmer: counting-aware network for handwritten mathematical expression recognition. In: Proceedings of the European conference on computer vision, pp 197–214
3. Zhao W, Gao L, Yan Z, Peng S, Du L, Zhang Z (2021) Handwritten mathematical expression recognition with bidirectionally trained transformer. In: Proceedings of the international conference on document analysis and recognition, pp 570–584
4. Bian X, Qin B, Xin X, Li J, Su X, Wang Y (2022) Handwritten mathematical expression recognition via attention aggregation based bi-directional mutual learning. In: Proceedings of the AAAI conference on artificial intelligence, pp 113–121
5. Lin T, Dollár P, Girshick RB, He K, Hariharan B, Belongie SJ (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 936–944 (2017)
6. Tian Z, Shen C, Chen H, He T (2019) Fcos: fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 9627–9636
7. Ohyama W, Suzuki M, Uchida S (2019) Detecting mathematical expressions in scientific document images using a u-net trained on a diverse dataset. IEEE Access 7:144030–144042
8. Mali P, Kukkadapu P, Mahdavi M, Zanibbi R (2020) Scanssd: scanning single shot detector for mathematical formulas in pdf document images. arXiv:2003.08005
9. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: single shot multibox detector. In: Proceedings of the European conference on computer vision
10. Duan K, Bai S, Xie L, Qi H, Huang Q, Tian Q (2019) Centernet: keypoint triplets for object detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 6569–6578
11. Zhang Z, Zhang C, Shen W, Yao C, Liu W, Bai X (2016) Multi-oriented text detection with fully convolutional networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4159–4167
12. Zhou X, Yao C, Wen H, Wang Y, Zhou S, He W, Liang J (2017) EAST: an efficient and accurate scene text detector. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2642–2651
13. He W, Zhang X, Yin F, Liu C (2017) Deep direct regression for multi-oriented scene text detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 745–753
14. Liao M, Shi B, Bai X, Wang X, Liu W (2017) Textboxes: a fast text detector with a single deep neural network. In: Proceedings of the thirty-first AAAI conference on artificial intelligence, pp 4161–4167
15. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. In: Proceedings of the advances in neural information processing systems, vol 28
16. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv:1804.02767
17. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: Proceedings of the European conference on computer vision, vol 12346, pp 213–229
18. Chen Q, Wang Y, Yang T, Zhang X, Cheng J, Sun J (2021) You only look one-level feature. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13039–13048
19. Zhang J, Du J, Zhang S, Liu D, Hu Y, Hu J, Wei S, Dai L (2017) Watch, attend and parse: an end-to-end neural network based approach to handwritten mathematical expression recognition. Pattern Recognit 71:196–206
20. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 770–778
21. Voigtlaender P, Doetsch P, Ney H (2016) Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In: Proceedings of the international conference on frontiers in handwriting recognition, pp 228–233
22. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
23. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Proc Adv Neural Inf Process Syst 30:6000–6010
24. Gers FA, Schmidhuber J (2000) Recurrent nets that time and count. In: Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks, pp 189–194
25. Liu X, Liang D, Yan S, Chen D, Qiao Y, Yan J (2018) FOTS: fast oriented text spotting with a unified network. In: In Proceedings of IEEE conference on computer vision and pattern recognition, pp 5676–5685
26. Jing L, Tian Y (2020) Self-supervised visual feature learning with deep neural networks: a survey. IEEE Trans Pattern Anal Mach Intell 43(11):4037–4058
27. He K, Fan H, Wu Y, Xie S, Girshick RB (2020) Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9726–9735
28. Chen T, Kornblith S, Norouzi M, Hinton GE (2020) A simple framework for contrastive learning of visual representations. In:

Proceedings of the 37th international conference on machine learning, vol 119, pp 1597–1607
29. Chen X, He K (2021) Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 15750–15758
30. Lin T, Goyal P, Girshick RB, He K, Dollár P (2020) Focal loss for dense object detection. IEEE Trans Pattern Anal Mach Intell 42(2):318–327
31. Parvaneh S, Rubin J, Rahman A, Conroy B, Babaeizadeh S (2017) Densely connected convolutional networks and signal quality analysis to detect atrial fibrillation using short single-lead ECG recordings. In: Computing in cardiology, CinC 2007, Rennes, September 24–27, 2017
32. Grill J-B, Strub F, Altché F, Tallec C, Richemond P, Buchatskaya E, Doersch C, Avila Pires B, Guo Z, Gheshlaghi Azar M et al (2020) Bootstrap your own latent—a new approach to self-supervised learning. Proc Adv Neural Inf Process Syst 33:21271–21284
33. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv:1609.04747
34. Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv:1212.5701