

# When Counting Meets HMER: Counting-Aware Network for Handwritten Mathematical Expression Recognition

Bohan Li<sup>1,2\*</sup>, Ye Yuan<sup>1\*</sup>, Ding kang Liang<sup>2</sup>, Xiao Liu<sup>1</sup>, Zhilong Ji<sup>1</sup>,  
Jinfeng Bai<sup>1</sup>, Wenyu Liu<sup>2</sup>, and Xiang Bai<sup>2‡</sup>

<sup>1</sup> Tomorrow Advancing Life

{yuanye\_phy}@hotmail.com, {jizhilong}@tal.com  
{ender.liux, jfbai.bit}@gmail.com

<sup>2</sup> Huazhong University of Science and Technology  
{bohan1024, dkliang, liuwy, xbai}@hust.edu.cn

**Abstract.** Recently, most handwritten mathematical expression recognition (HMER) methods adopt the encoder-decoder networks, which directly predict the markup sequences from formula images with the attention mechanism. However, such methods may fail to accurately read formulas with complicated structure or generate long markup sequences, as the attention results are often inaccurate due to the large variance of writing styles or spatial layouts. To alleviate this problem, we propose an unconventional network for HMER named Counting-Aware Network (CAN), which jointly optimizes two tasks: HMER and symbol counting. Specifically, we design a weakly-supervised counting module that can predict the number of each symbol class without the symbol-level position annotations, and then plug it into a typical attention-based encoder-decoder model for HMER. Experiments on the benchmark datasets for HMER validate that both joint optimization and counting results are beneficial for correcting the prediction errors of encoder-decoder models, and CAN consistently outperforms the state-of-the-art methods. In particular, compared with an encoder-decoder model for HMER, the extra time cost caused by the proposed counting module is marginal. The source code is available at <https://github.com/LBH1024/CAN>.

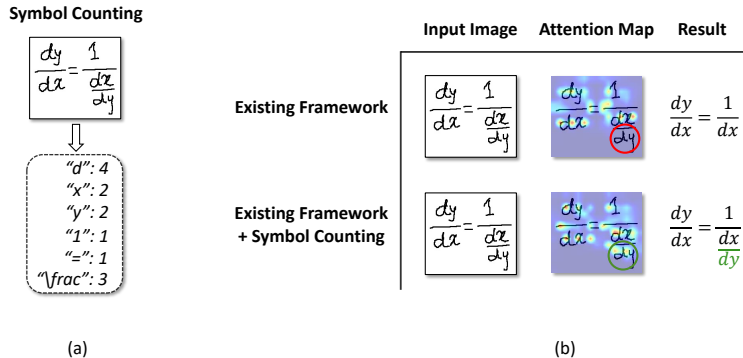
**Keywords:** Handwritten mathematical expression recognition · Attention mechanism · Counting

## 1 Introduction

Handwritten mathematical expression recognition (HMER) is an important task of document analysis, which has broad applications including assignment grading, digital library service, and office automation. Despite the great successes of the current OCR systems, HMER still remains a very challenging problem due to the complex structures of formulas or irregular writings.

---

\* Authors contribute equally. ‡ Corresponding author.



**Fig. 1.** (a) Illustration of the symbol counting task. (b) Comparison between the existing framework (e.g., DWAP [40]) and the proposed framework (CAN). By visualizing the attention map, we can observe that the existing framework misses the denominator “ $dy$ ” while our CAN correctly locate it after using symbol counting.

Encoder-decoder architectures are extensively used in the recent HMER approaches [1, 31, 40], which formulate HMER as an image-to-sequence translation problem. Given a handwritten formula, such methods predict its corresponding markup sequence (e.g., LaTeX) with the attention mechanism. However, encoder-decoder models often cannot guarantee the accuracy of attention, especially when the structure of a handwritten formula is complicated or the markup sequence is long.

In this paper, we propose an unconventional method for improving the robustness of the encoder-decoder models for HMER. We argue that counting and HMER are two complementary tasks, and using counting can improve the performance of HMER. In this community, object counting [12, 15] has been intensively studied, but has seldom been applied in the OCR area. Our intuition includes two aspects: 1) symbol counting (as illustrated in Fig. 1(a)) is able to provide the symbol-level position information, which can make the attention results more accurate. 2) The counting results, representing the number of each symbol class, can serve as additional global information to promote recognition accuracy.

Specifically, we design a weakly-supervised counting module named MSCM, which can be easily plugged into existing encoder-decoder networks and optimized jointly in an end-to-end manner. With this counting module, an encoder-decoder model can be better aware of each symbol’s position, as shown in Fig. 1(b). It is worth noticing that the proposed counting module just needs original HMER annotations (LaTeX sequences) without extra labeling work. We combine our counting module with a typical encoder-decoder model (e.g., DWAP [40]), proposing a unified network for HMER named Counting-Aware Network (CAN). We test it on the benchmark datasets and observe that both HMER and symbol counting gain obvious and consistent performance improvement. In particular,

compared with the original model, the extra time cost brought by MSCM is marginal.

In summary, the main contributions of this paper are two-fold. 1) To the best of our knowledge, we are the first to bring symbol counting into HMER and reveal the relevance and the complementarity of HMER and symbol counting. 2) We propose a new method that jointly optimizes symbol counting and HMER, which consistently improves the performance of the encoder-decoder models for HMER.

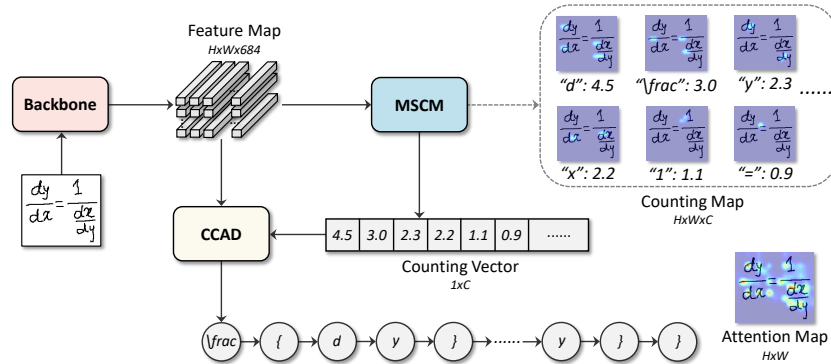
To be specific about the performance, with adopting DWAP [40] as the baseline network, our method achieves state-of-the-art (SOTA) recognition accuracy on the widely-used CROHME dataset (57.00% on CROHME 2014, 56.06% on CROHME 2016, 54.88% on CROHME 2019). Moreover, with adopting the latest SOTA method ABM [1] as the baseline network, CAN achieves new SOTA results (57.26% on CROHME 2014, 56.15% on CROHME 2016, 55.96% on CROHME 2019). This indicates that our method can be generalized to various existing encoder-decoder models for HMER and boost their performance.

## 2 Related Work

### 2.1 HMER

Traditional HMER methods usually take a three-step approach: a symbol segmentation step, a symbol recognition step, and a grammar-guided structure analysis step. Classic classification techniques such as HMM [8, 11, 29], Elastic Matching [2, 26] and Support Vector Machines [10] are mainly used in the recognition step. In the structure analysis step, formal grammars are elaborately designed to model the 2D and syntactic structures of formulas. Lavirotte *et al.* [13] propose to use graph grammar to recognize mathematical expression. Chan *et al.* [3] incorporate correction mechanism into a parser based on definite clause grammar (DCG). However, limited feature learning ability and complex grammar rules make the traditional methods far to meet real-world application.

Recently, deep learning has rapidly boosted the performance of HMER. The mainstream framework is the encoder-decoder network [1, 5, 24, 30, 31, 40, 42, 43, 46]. Deng *et al.* [5] first apply an attention-based encoder-decoder model in HMER, inspired by its success in image caption task [34]. Zhang *et al.* [43] also present a similar model named WAP. In their model, they apply a FCN as the encoder and utilize the coverage attention, which is the sum of all past attention weights, to alleviate the lack of coverage problem [25]. Wu *et al.* [30, 31] focus on the pair-wise adversarial learning strategy to improve the recognition accuracy. Later, Zhang *et al.* [42] devise a tree-based decoder to parse formulas. At each step, a parent and child node pair is generated and the relation between parent node and child node reflects the structure type. Bi-directional learning has been proven effective to improve model recognition performance [23]. Zhao *et al.* [46] design a bi-directionally trained transformer framework and Bian *et al.* [1] propose an bi-directional mutual learning network. They further prove bi-directional learning can also significantly improve the HMER performance.



**Fig. 2.** Structure of the proposed CAN, which consists of a backbone network, a multi-scale counting Module (MSCM) and a counting-combined attentional decoder (CCAD).

## 2.2 Object Counting

Object counting can be roughly divided into two categories, detection-based and regression-based. The detection-based methods [17, 21] obtain the number by detecting each instance. The regression-based methods [15, 33] learn to count by regressing a density map, and the predicted count equals the integration of the density map. To improve the counting accuracy, multi-scale strategy [44], attention mechanism [39] and perspective information [35] are widely adopted in the regression-based methods. Nevertheless, both detection-based and density map regression-based methods need the object position annotations (fully-supervised), such as box-level [17, 21] and point-level [15, 33, 44] annotations. To relieve the expensive and laborious labeling work, several approaches [27, 36] that only use count-level annotations (weakly-supervised) are proposed. And they find that the visualized feature map can accurately reflect the object regions. Different from most of the previous counting modules that are category specifically (e.g., crowd counting), our counting module is designed for multi-class object counting since formulas usually contains various symbols. In the OCR area, Xie *et al.* [32] propose a counting-based loss function mainly designed for scene texts (words or text-lines), while our model can exploit the counting information of more complicated texts (e.g., mathematical expressions) at both the feature level and the loss level.

## 3 Methodology

### 3.1 Overview

As shown in Fig. 2, our Counting-Aware Network (CAN) is a unified end-to-end trainable framework that comprises a backbone, a multi-scale counting module (MSCM) and a counting-combined attentional decoder (CCAD). Following DWAP [40], we apply DenseNet [9] as the backbone. Given a gray-scale

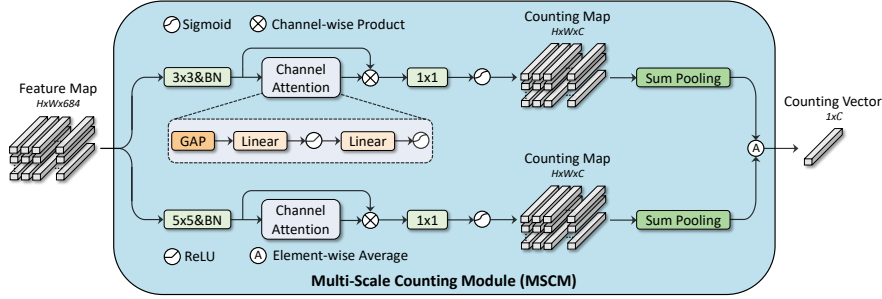


Fig. 3. Structure of the proposed multi-scale counting module (MSCM).

image  $\mathcal{X} \in \mathbb{R}^{H' \times W' \times 1}$ , the backbone is first used to extract 2D feature map  $\mathcal{F} \in \mathbb{R}^{H \times W \times 684}$ , where  $\frac{H'}{H} = \frac{W'}{W} = 16$ . The feature map  $\mathcal{F}$  will be used by both the MSCM and the CCAD. The counting module MSCM is used to predict the number of each symbol class and generate the 1D counting vector  $\mathcal{V}$  that represents the counting results. The feature map  $\mathcal{F}$  and the counting vector  $\mathcal{V}$  will be fed into the CCAD to get predicted output.

### 3.2 Multi-Scale Counting Module

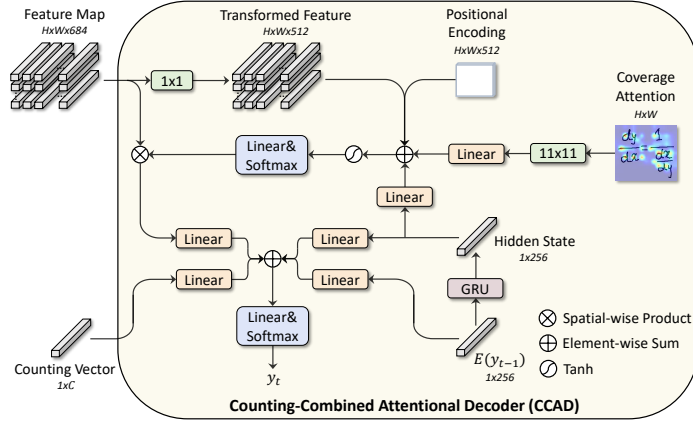
In this part, we present the detail of the proposed multi-scale counting module (MSCM), which is designed to predict the number of each symbol class. Specifically, as depicted in Fig. 3, MSCM consists of multi-scale feature extraction, channel attention and sum-pooling operator. Formula images usually contain various sizes of symbols due to different writing habits. Single kernel size can not effectively handle the scale variations. To this end, we first utilize two parallel convolution branches to extract multi-scale features by using different kernel sizes (set to  $3 \times 3$  and  $5 \times 5$ ). Following the convolution layer, the channel attention [7] is adopted to enhance the feature information further. Here, we choose one of the branches for simple illustration. Let us denote  $\mathcal{H} \in \mathbb{R}^{H \times W \times C'}$  as the extracted feature map from the convolution ( $3 \times 3$  or  $5 \times 5$ ) layer. The enhanced feature  $\mathcal{S}$  can be written as:

$$\mathcal{Q} = \sigma(W_1(G(\mathcal{H})) + b_1), \quad (1)$$

$$\mathcal{S} = \mathcal{Q} \otimes g(W_2\mathcal{Q} + b_2), \quad (2)$$

where  $G$  is the global average pooling.  $\sigma$  and  $g(\cdot)$  refer to ReLU and sigmoid function, respectively.  $\otimes$  denotes channel-wise product and  $W_1, W_2, b_1, b_2$  are trainable weights.

After getting the enhanced feature  $\mathcal{S}$ , we use a  $1 \times 1$  convolution to reduce the channel number from  $C'$  to  $C$ , where  $C$  is the number of symbol classes.



**Fig. 4.** Structure of the proposed counting-combined attentional decoder (CCAD).

Ideally, the symbol counting result should mainly calculate from the foreground (symbols), i.e., the response of the background should be close to zero. Thus, following the  $1 \times 1$  convolution, we utilize a sigmoid function to yield the value in a range of  $(0,1)$  to generate counting map  $\mathcal{M} \in \mathbb{R}^{H \times W \times C}$ . For each  $\mathcal{M}_i \in \mathbb{R}^{H \times W}$ , it is supposed to effectively reflect the position of the  $i$ -th symbol class, as shown in Fig. 2. In this sense, each  $\mathcal{M}_i$  is actually a pseudo density map, and we can utilize sum-pooling operator to obtain counting vector  $\mathcal{V} \in \mathbb{R}^{1 \times C}$ :

$$\mathcal{V}_i = \sum_{p=1}^H \sum_{q=1}^W M_{i,pq} \quad (3)$$

Here,  $\mathcal{V}_i \in \mathbb{R}^{1 \times 1}$  is the predicted count of the  $i$ -th class symbol. It is noteworthy that the feature maps of different branches contain different scale information and are highly complementary. Thus, we combine the complementary counting vectors and use the average operator to generate the final result  $\mathcal{V}^f \in \mathbb{R}^{1 \times C}$ , which is then fed into the decoder CCAD.

### 3.3 Counting-Combined Attentional Decoder

The structure of our counting-combined attentional decoder (CCAD) is shown in Fig. 4. Given the 2D feature map  $\mathcal{F} \in \mathbb{R}^{H \times W \times 684}$ , we first use a  $1 \times 1$  convolution to change the number of channel and get transformed feature  $\mathcal{T} \in \mathbb{R}^{H \times W \times 512}$ . Then, to enhance model’s awareness of spatial position, we use a fixed absolute encoding  $\mathcal{P} \in \mathbb{R}^{H \times W \times 512}$  to represent different spatial positions in  $\mathcal{T}$ . Specifically, we adopt the spatial positional encoding [20], which independently uses sine and cosine functions with different frequencies for both spatial coordinates.

During the decoding process, when decoding at step  $t$ , we pass the embedding of symbol  $y_{t-1}$  into a GRU cell [4] to get a hidden state  $h_t \in \mathbb{R}^{1 \times 256}$ . With the

transformed feature  $\mathcal{T}$  and the spatial encoding  $\mathcal{P}$ , we can then get the attention weights  $\alpha_t \in \mathbb{R}^{H \times W}$  as follows:

$$e_t = w^T \tanh(\mathcal{T} + \mathcal{P} + W_a \mathcal{A} + W_h h_t) + b, \quad (4)$$

$$\alpha_{t,ij} = \exp(e_{t,ij}) / \sum_{p=1}^H \sum_{q=1}^W e_{t,pq}, \quad (5)$$

where  $w$ ,  $b$ ,  $W_a$ ,  $W_h$  are trainable weights and coverage attention  $\mathcal{A}$  is the sum of all past attention weights.

Applying spatial-wise product to the attention weights  $\alpha_t$  and the feature map  $\mathcal{F}$ , we can get context vector  $\mathcal{C} \in \mathbb{R}^{1 \times 256}$ . In most of the previous HMER methods, they predict  $y_t$  only using the context vector  $\mathcal{C}$ , the hidden state  $h_t$  and the embedding  $E(y_{t-1})$ . Actually,  $\mathcal{C}$  just corresponds to a local region of the feature map  $\mathcal{F}$ . And we argue that  $h_t$  and  $E(y_{t-1})$  also lack global information. Considering that the counting vector  $\mathcal{V}$  is calculated from a global counting perspective, which can serve as additional global information to make the prediction more accurate, we combine them together to predict  $y_t$  as follows:

$$p(y_t) = \text{softmax}(w_o^T (W_c \mathcal{C} + W_v \mathcal{V} + W_t h_t + W_e E)) + b_o, \quad (6)$$

$$y_t \sim p(y_t), \quad (7)$$

where  $w_o$ ,  $b_o$ ,  $W_c$ ,  $W_v$ ,  $W_t$ ,  $W_e$  are trainable weights.

### 3.4 Loss Function

The overall loss function consists of two parts and is defined as follows:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{counting}, \quad (8)$$

where  $\mathcal{L}_{cls}$  is a common-used cross entropy classification loss of the predicted probability  $p(y_t)$  with respect to its ground-truth. Denoting the counting ground truth of each symbol class as  $\hat{\mathcal{V}}$ ,  $\mathcal{L}_{counting}$  is a smooth  $L1$  [22] regression loss defined as follows:

$$\mathcal{L}_{counting} = \text{smooth}_{L1}(\mathcal{V}, \hat{\mathcal{V}}) \quad (9)$$

## 4 Experiments

### 4.1 Datasets

**CROHME Dataset** [18] is the most widely-used public dataset in the field of HMER, which is from the competition on recognition of online handwritten mathematical expressions (CROHME). The CROHME training set contains 8836 handwritten mathematical expressions, and there are three testing sets:

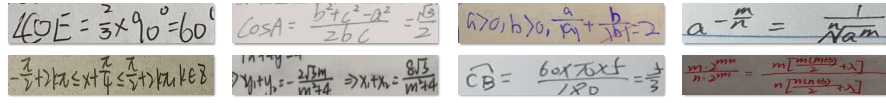


Fig. 5. Some example images from the HME100K dataset.

CROHME 2014, 2016, 2019 with 986, 1147, and 1199 handwritten mathematical expressions, respectively. The number of symbol classes  $C$  is 111, including “*sos*” and “*eos*”. In the CROHME dataset, each handwritten mathematical expression is stored in InkML format, which records the trajectory coordinates of handwritten strokes. We convert the handwritten stroke trajectory information in the InkML files into image format for training and testing.

**HME100K Dataset** [37] is a real scene handwritten mathematical expression dataset, consisting of 74,502 images for training and 24,607 images for testing. The number of symbol classes  $C$  is 249 including “*sos*” and “*eos*”. These images are from tens of thousands of writers, mainly captured by cameras. Consequently, HME100K is more authentic and realistic with variations in color, blur, and complicated background. Some example images are shown in Fig. 5.

## 4.2 Implementation Details

The proposed CAN is implemented in PyTorch. We use a single Nvidia Tesla V100 with 32GB RAM to train our model with batch size 8 and the Adadelta optimizer [38]. The learning rate starts from 0 and monotonously increases to 1 at the end of the first epoch and decays to 0 following the cosine schedules [45]. For the CROHME dataset, the total training epoch is set to 240, and we separately present the results with and without data augmentation. Compared with previous methods, we use different data augmentation (rotation, affine, perspective, erosion and dilation) to explore the ability of our method. For the HME100K dataset, the total training epoch is set to 30 without data augmentation.

It is noteworthy that when counting the symbols in the handwritten mathematical expression, six classes of symbols are ignored by assigning their counting ground truth as zero because they are invisible: “*sos*”, “*eos*”, “ $\wedge$ ”, “ $-$ ”, “ $\{$ ”, “ $\}$ ”. Counting these symbols will confuse the model and bring lower accuracy.

## 4.3 Evaluation Metrics

**Expression recognition.** Expression recognition rate (ExpRate), defined as the percentage of correctly recognized expressions, is used to evaluate the performance of different methods on mathematical expression recognition. Moreover,  $\leq 1$  and  $\leq 2$  are also used, indicating that the expression recognition rate is tolerable at most one or two symbol-level errors.

**Symbol counting.** The mean absolute error (MAE) and the mean squared error (MSE) are the primary metrics in the object counting task. In our multi-class symbol counting task, we use MAE and MSE to evaluate the counting



**Table 1.** Results on the CROHME dataset. \* indicates using stoke trajectory coordinates annotations. † indicates our reproduced result. CAN-DWAP and CAN-ABM represent using DWAP and ABM as the baseline respectively. Note that we use different data augmentation than previous methods. Our intention is to show that even with data augmentation, our counting module can still stably improve existing HMER methods’ performance.

Method	CROHME 2014			CROHME 2016			CROHME 2019		
	ExpRate†	≤ 1 †	≤ 2 †	ExpRate†	≤ 1 †	≤ 2 †	ExpRate†	≤ 1 †	≤ 2 †
Without data augmentation									
UPV [18]	37.22	44.22	47.26	-	-	-	-	-	-
TOKYO [19]	-	-	-	43.94	50.91	53.70	-	-	-
PAL [30]	39.66	56.80	65.11	-	-	-	-	-	-
WAP [43]	46.55	61.16	65.21	44.55	57.10	61.55	-	-	-
PAL-v2 [31]	48.88	64.50	69.78	49.61	64.08	70.27	-	-	-
TAP [41]*	48.47	63.28	67.34	44.81	59.72	62.77	-	-	-
DLA [14]	49.85	-	-	47.34	-	-	-	-	-
DWAP [40]	50.10	-	-	47.50	-	-	-	-	-
DWAP-TD [42]	49.10	64.20	67.80	48.50	62.30	65.30	51.40	66.10	69.10
DWAP-MSA [40]	52.80	68.10	72.00	50.10	63.80	67.40	47.70	59.50	63.30
WS-WAP [24]	53.65	-	-	51.96	64.34	70.10	-	-	-
MAN [28]*	54.05	68.76	72.21	50.56	64.78	67.13	-	-	-
BTTR [46]	53.96	66.02	70.28	52.31	63.90	68.61	52.96	65.97	69.14
ABM [1]	56.85	73.73	81.24	52.92	69.66	78.73	53.96	71.06	78.65
DWAP (baseline)†	51.48	67.01	73.30	50.65	63.30	70.88	50.04	65.39	69.39
CAN-DWAP (ours)	<b>57.00</b>	74.21	80.61	<b>56.06</b>	71.49	79.51	<b>54.88</b>	71.98	79.40
ABM (baseline)†	56.04	73.10	79.90	53.36	70.01	78.12	53.71	71.23	78.23
CAN-ABM (ours)	<b>57.26</b>	74.52	82.03	<b>56.15</b>	72.71	80.30	<b>55.96</b>	72.73	80.57
With data augmentation									
Li <i>et al.</i> [16]	56.59	69.07	75.25	54.58	69.31	73.76	-	-	-
Ding <i>et al.</i> [6]	58.72	-	-	57.72	70.01	76.37	61.38	75.15	80.23
DWAP (baseline)†	57.97	73.81	79.19	55.97	71.40	79.86	56.05	72.23	79.15
CAN-DWAP (ours)	<b>65.58</b>	77.36	83.35	<b>62.51</b>	74.63	82.48	<b>63.22</b>	78.07	82.49
ABM (baseline)†	63.76	76.35	83.05	60.86	73.93	81.17	62.22	77.23	81.90
CAN-ABM (ours)	<b>65.89</b>	77.97	84.16	<b>63.12</b>	75.94	82.74	<b>64.47</b>	78.73	82.99

performance for each formula image, and then average the counting results of all formula images to get  $MAE_{Ave}$  and  $MSE_{Ave}$ :

$$MAE = \frac{1}{C} \sum_{i=1}^C |\mathcal{V}_i - \hat{\mathcal{V}}_i|, \quad MSE = \sqrt{\frac{1}{C} \sum_{i=1}^C |\mathcal{V}_i - \hat{\mathcal{V}}_i|^2}, \quad (10)$$

$$MAE_{Ave} = \frac{1}{N} \sum_{i=1}^N MAE_i, \quad MSE_{Ave} = \frac{1}{N} \sum_{i=1}^N MSE_i, \quad (11)$$

where  $C$  denotes the number of symbol classes,  $N$  is the number of images in the testing set,  $\mathcal{V}_i$  and  $\hat{\mathcal{V}}_i$  are the predicted count and its corresponding ground truth of a symbol class respectively.

#### 4.4 Comparison with State-of-the-Art

To demonstrate the superiority of our method, we compare it with previous state-of-the-art (SOTA) methods. Table 1 shows the expression recognition rate

Input Image	DWAP (Baseline)	CAN-DWAP (Ours)
	$\log g$	$\log$
	$F(b) - F(G)$	$F(b) - F(a)$
	$\sum_{n=1}^{\infty} \frac{\cos \pi}{n}$	$\sum_{n=1}^{\infty} \frac{\cos \pi n}{n}$
	$x^5 + y^5 - xy + 1 = 0$	$x^5 + y^5 - 5xy + 1 = 0$
	$\sum_{n=1}^{1000} (0001-n)^{-2}$	$\sum_{n=1}^{1000} (10001-n)^{-2}$

Fig. 6. Some recognition cases of DWAP and CAN-DWAP.

(ExpRate) on the CROHME dataset. Most of the previous methods do not use data augmentation, so we mainly focus on the results produced without data augmentation.

As shown in Table 1, with adopting DWAP [40] as the baseline, CAN-DWAP achieves SOTA results on CROHME 2014, CROHME 2016, CROHME 2019 and outperforms the latest SOTA method ABM [1] on CROHME 2016 by a significant margin of 3.14%. Fig. 6 shows some qualitative recognition results of DWAP and CAN-DWAP. We can observe that our method is less likely to miss symbols or predict redundant symbols.

To further verify the effectiveness of our method, we reproduce the latest SOTA method ABM [1] and adopt it as our baseline to construct CAN-ABM. As shown in Table 1, CAN-ABM outperforms its baseline and achieves new SOTA results. This indicates that our method can be generalized to various existing encoder-decoder models for HMER and boost their performance.

#### 4.5 Results on the HME100K Dataset

Although the CROHME dataset has been widely used and has great influence in the field of HMER, its small size limits the performance of different methods. Hence, we further evaluate our method on the HME100K dataset, which is nearly ten times larger than the CROHME dataset and has more variations in color, blur, and background. The quantitative results are listed in Table 2, CAN-DWAP and CAN-ABM largely outperform their baseline DWAP [40] and ABM [1] respectively.

#### 4.6 Inference Speed

To explore the efficiency of our proposed method, we evaluate its speed on the HME100K dataset with a single Nvidia Tesla V100. As shown in Table 3, compared with the baseline model, the extra parameters and FLOPs are mainly

**Table 2.** Results on the HME100K dataset. † indicates our reproduced result. CAN-DWAP and CAN-ABM represent using DWAP and ABM as the baseline respectively.

Method	HME100K		
	ExpRate†	$\leq 1$ †	$\leq 2$ †
DWAP-TD [42]†	62.60	79.05	85.67
DWAP [40] (baseline)†	61.85	70.63	77.14
CAN-DWAP (ours)	67.31	82.93	89.17
ABM [1] (baseline)†	65.93	81.16	87.86
CAN-ABM (ours)	<b>68.09</b>	<b>83.22</b>	<b>89.91</b>

**Table 3.** Comparison on parameters, FLOPs, and FPS.

Method	HME100K			
	#Params	Input size	FLOPs	FPS
DWAP [40] (baseline)	4.7M	(1,1,120,800)	9.7G	23.3
CAN-DWAP (ours)	17.0M	(1,1,120,800)	14.7G	21.7

brought by the counting module’s two convolution layers with kernels of sizes  $3 \times 3$  and  $5 \times 5$ . As to the inference speed, the extra time cost brought by the counting module is marginal.

#### 4.7 Ablation Study

**Component Analysis.** In our method, symbol counting serves as an auxiliary task and influences the feature learning together with the primary task HMER through joint optimization. Meanwhile, adding counting vector during the decoding process also has an impact on the performance. So, to verify the effectiveness of the three components: positional encoding, joint optimization, and counting vector, we conduct experiments and the results are listed in Table 4. We can observe that both joint optimization and counting vector can boost the performance to a certain degree, and adding positional encoding can also slightly improve the recognition accuracy.

**Impact of Convolution Kernel in Counting Module.** In our counting module MSCM, we adopt a multi-scale strategy by using convolution layer with different sizes of kernels ( $3 \times 3$  and  $5 \times 5$ ). To explore the impact of different convolution kernels, we conduct experiments on CROHME 2014 with using different sizes of convolution kernels. As shown in Table 5, using  $3 \times 3$  and  $5 \times 5$  convolution kernels together achieves the best results (57.00% ExpRate, 0.033  $MAE_{Ave}$  and 0.037  $MSE_{Ave}$ ). Using either  $3 \times 3$  or  $5 \times 5$  convolution kernel will get lower counting accuracy and lower ExpRate. We think this phenomenon indicates that multi-scale information obtained with different kinds of convolution kernels can help the counting module better tackle the size variations.

**Impact of Counting Vector on HMER.** To explore the impact of counting vector, we use the ground truth of counting vector and add different random disturbances to it (e.g., randomly add or subtract 1) so that we can get several counting vectors with different  $MAE_{Ave}$  and  $MSE_{Ave}$ . By providing these counting vectors to the decoder during training and testing, we conduct several

**Table 4.** Ablation study of different components.

Method	CROHME 2014			CROHME 2016			CROHME 2019		
	ExpRate $\uparrow$	$\leq 1 \uparrow$	$\leq 2 \uparrow$	ExpRate $\uparrow$	$\leq 1 \uparrow$	$\leq 2 \uparrow$	ExpRate $\uparrow$	$\leq 1 \uparrow$	$\leq 2 \uparrow$
DWAP [40] (baseline)	51.48	67.01	73.30	50.65	63.30	70.88	50.04	65.39	69.39
+ Positional encoding	51.88	68.12	74.21	51.00	64.06	71.37	50.96	66.14	70.48
+ Joint optimization	55.23	72.18	78.17	54.11	68.00	76.37	53.13	69.89	76.00
+ Counting vector	<b>57.00</b>	<b>74.21</b>	<b>80.61</b>	<b>56.06</b>	<b>71.49</b>	<b>79.51</b>	<b>54.88</b>	<b>71.98</b>	<b>79.40</b>

**Table 5.** Ablation study of different convolution kernels in counting module.

Method	CROHME 2014				
	ExpRate $\uparrow$	$\leq 1 \uparrow$	$\leq 2 \uparrow$	$MAE_{Ave} \downarrow$	$MSE_{Ave} \downarrow$
CAN-DWAP ( $3 \times 3$ )	54.92	71.26	78.07	0.048	0.046
CAN-DWAP ( $5 \times 5$ )	55.53	71.88	78.58	0.044	0.043
CAN-DWAP ( $3 \times 3$ & $5 \times 5$ )	<b>57.00</b>	<b>74.21</b>	<b>80.61</b>	<b>0.033</b>	<b>0.037</b>

**Table 6.** Ablation study of different counting vectors. \* indicates adding random disturbance to counting vector. The latter counting GT with \* is added with more disturbances than the former one.

Method	CROHME 2014				
	ExpRate $\uparrow$	$\leq 1 \uparrow$	$\leq 2 \uparrow$	$MAE_{Ave} \downarrow$	$MSE_{Ave} \downarrow$
CAN-DWAP	57.00	74.21	80.61	0.033	0.037
CAN-DWAP (counting GT)*	58.28	74.92	81.02	0.027	0.025
CAN-DWAP (counting GT)*	60.10	76.04	81.73	0.019	0.016
CAN-DWAP (counting GT)	62.44	76.14	82.23	0.000	0.000

experiments and the results are shown in Table 6. When using the ground truth of counting vector, the ExpRate on CROHME 2014 reaches 62.44%. As more disturbances are added, the counting vector becomes more inaccurate, and the ExpRate drops consequently.

**Impact of HMER on Symbol Counting.** Through joint optimization, symbol counting can promote the performance of HMER. To find out whether HMER can also promote the performance of symbol counting, we train CAN only with the symbol counting task and compare it with CAN trained with two tasks. As shown in Table 7, HMER can boost the performance of symbol counting with improving  $MAE_{Ave}$  by 31.25% and  $MSE_{Ave}$  by 15.91%.

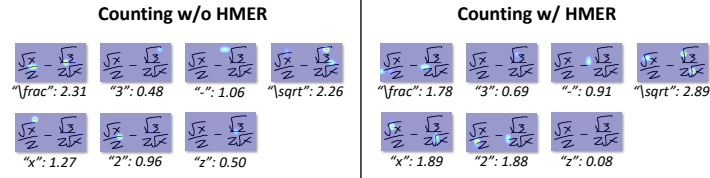
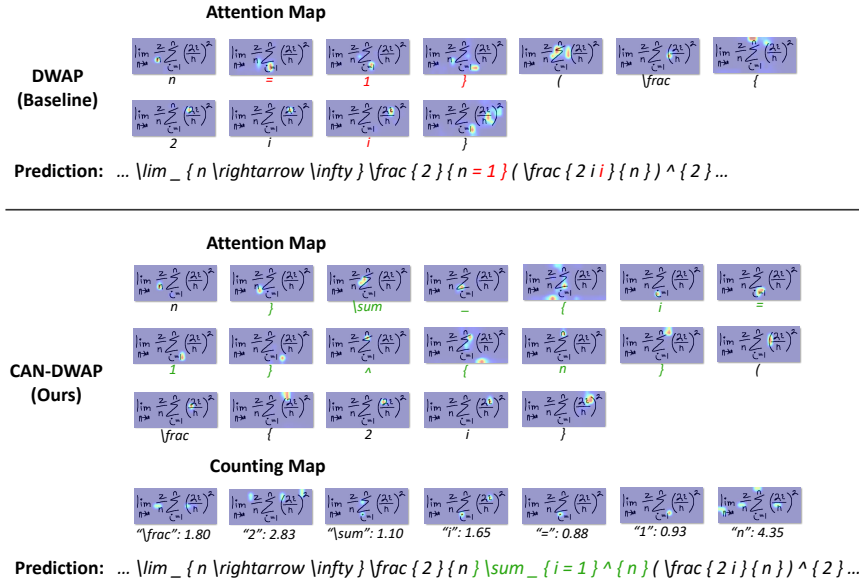
Some visual results are shown in Fig. 7. We can observe that when training only with the symbol counting task, some symbols are wrongly located (e.g., “\_”) or partially counted (e.g., “2”). Counting with the HMER task can alleviate this problem by providing context-aware information, which is gained through the context-aware decoding process in the decoder CCAD.

#### 4.8 Case Study with Maps

In this part, we choose a typical example to visualize its counting map from the counting module and its attention map from the decoder. As illustrated in Fig. 8, after predicting the symbol “n”, DWAP misses the symbol “ $\sum$ ” and the symbol “i” and directly predicts the symbol “=”. The missing symbol “ $\sum$ ” is noticed later by the model when predicting the symbol “(” but the mistake has

**Table 7.** Ablation study of HMER’s impact on symbol counting.

Method	CROHME 2014	
	$MAE_{Ave} \downarrow$	$MSE_{Ave} \downarrow$
Counting w/o HMER	0.048	0.044
Counting w HMER	<b>0.033</b>	<b>0.037</b>

**Fig. 7.** Counting map generated with and without HMER task.**Fig. 8.** Counting map and attention map of DWAP and CAN-DWAP.

already happened in this sequential decoding process. A redundant symbol “ $i$ ” is also wrongly predicted, and the attention map shows that this mistake is due to the model’s repeated attention on the symbol “ $i$ ”.

In contrast, our method CAN-DWAP predicts the formula correctly. From the counting map, we can see that almost all symbols are accurately located (note that we do not use symbol-level position annotations). And the predicted count of each symbol class, which is calculated by summing each counting map, is very close to its ground truth. These phenomenons demonstrate that by counting

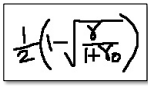
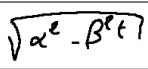
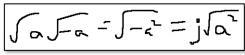
Input Image	CAN-DWAP (Ours)	Ground Truth
	$\frac{1}{\sqrt{a + \sqrt{1 + \sqrt{\gamma}}}}$	$\frac{1}{\sqrt{a + \sqrt{1 + \gamma_0}}}$
	$\sqrt{\alpha^2 - \beta^p t}$	$\sqrt{\alpha^2 - \beta^2 t}$
	$\sqrt{a} \sqrt{-a} = \sqrt{-a^2} = j \sqrt{a^2}$	$\sqrt{a} \sqrt{-a} = \sqrt{-a^2} = j \sqrt{a^2}$

Fig. 9. Some failure cases of our CAN-DWAP.

each symbol class, the model becomes more aware of each symbol, especially their positions. As a result, the model has more accurate attention results (seen from the attention map) during the decoding process and is less likely to miss or predict redundant symbols.

#### 4.9 Limitation

Despite the significant performance improvement brought by symbol counting, the variations in writing styles still cause some recognition problems and cannot be solved very well with symbol counting, as shown in Fig. 9. Moreover, since we do not explicitly model structure grammar, our method may make some mistakes when extreme fine structure perception ability is needed.

## 5 Conclusion

In this paper, we design a counting module MSCM, which can perform symbol counting just relying on the original HMER annotations (LaTeX sequences). By plugging this counting module into an attention-based encoder-decoder network, we propose an unconventional end-to-end trainable network for HMER named CAN, which jointly optimizes HMER and symbol counting. Experiments on the benchmark datasets for HMER validate three main conclusions. 1) Symbol counting can consistently improve the performance of the encoder-decoder models for HMER. 2) Both joint optimization and counting results contribute to this improvement. 3) HMER can also increase the accuracy of symbol counting through joint optimization.

## Acknowledgements

This work was done when Bohan Li was an intern at Tomorrow Advancing Life, and was supported in part by the National Natural Science Foundation of China 61733007 and the National Key R&D Program of China under Grant No. 2020AAA0104500.

## References

1. Bian, X., Qin, B., Xin, X., Li, J., Su, X., Wang, Y.: Handwritten mathematical expression recognition via attention aggregation based bi-directional mutual learning. In: Proc. of the AAAI Conf. on Artificial Intelligence. pp. 113–121 (2022)
2. Chan, K.F., Yeung, D.Y.: Elastic structural matching for online handwritten alphanumeric character recognition. In: Proc. of Intl. Conf. on Pattern Recognition. vol. 2, pp. 1508–1511 (1998)
3. Chan, K.F., Yeung, D.Y.: Error detection, error correction and performance evaluation in on-line mathematical expression recognition. *Pattern Recognition* **34**(8), 1671–1684 (2001)
4. Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing (2014)
5. Deng, Y., Kanervisto, A., Ling, J., Rush, A.M.: Image-to-markup generation with coarse-to-fine attention. In: Proc. of Intl. Conf. on Machine Learning. pp. 980–989 (2017)
6. Ding, H., Chen, K., Huo, Q.: An encoder-decoder approach to handwritten mathematical expression recognition with multi-head attention and stacked decoder. In: Proc. of International Conference on Document Analysis and Recognition. pp. 602–616 (2021)
7. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018)
8. Hu, L., Zanibbi, R.: Hmm-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature. In: Proc. of International Conference on Document Analysis and Recognition. pp. 457–462 (2011)
9. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition. pp. 4700–4708 (2017)
10. Keshari, B., Watt, S.: Hybrid mathematical symbol recognition using support vector machines. In: Proc. of International Conference on Document Analysis and Recognition. vol. 2, pp. 859–863 (2007)
11. Kosmala, A., Rigoll, G., Lavirotte, S., Pottier, L.: On-line handwritten formula recognition using hidden markov models and context dependent graph grammars. In: Proc. of International Conference on Document Analysis and Recognition. pp. 107–110 (1999)
12. Laradji, I.H., Rostamzadeh, N., Pinheiro, P.O., Vazquez, D., Schmidt, M.: Where are the blobs: Counting by localization with point supervision. In: Proc. of European Conference on Computer Vision. pp. 547–562 (2018)
13. Lavirotte, S., Pottier, L.: Mathematical formula recognition using graph grammar. In: Document Recognition V. vol. 3305, pp. 44–52 (1998)
14. Le, A.D.: Recognizing handwritten mathematical expressions via paired dual loss attention network and printed mathematical expressions. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition Workshops. pp. 566–567 (2020)
15. Li, Y., Zhang, X., Chen, D.: CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition (2018)

16. Li, Z., Jin, L., Lai, S., Zhu, Y.: Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. In: Proc. of International Conference on Frontiers in Handwriting Recognition. pp. 175–180 (2020)
17. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: Proc. of European Conference on Computer Vision. pp. 21–37 (2016)
18. Mouchere, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). In: Proc. of International Conference on Frontiers in Handwriting Recognition. pp. 791–796 (2014)
19. Mouchère, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions. In: Proc. of International Conference on Frontiers in Handwriting Recognition. pp. 607–612 (2016)
20. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: Proc. of Intl. Conf. on Machine Learning. pp. 4055–4064 (2018)
21. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Proc. of Advances in Neural Information Processing Systems **28** (2015)
22. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(06), 1137–1149 (2017)
23. Shi, B., Yang, M., Wang, X., Lyu, P., Yao, C., Bai, X.: Aster: An attentional scene text recognizer with flexible rectification. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(9), 2035–2048 (2018)
24. Truong, T.N., Nguyen, C.T., Phan, K.M., Nakagawa, M.: Improvement of end-to-end offline handwritten mathematical expression recognition by weakly supervised learning. In: Proc. of International Conference on Frontiers in Handwriting Recognition. pp. 181–186 (2020)
25. Tu, Z., Lu, Z., Liu, Y., Liu, X., Li, H.: Modeling coverage for neural machine translation. In: Proc. of the Association for Computational Linguistics. pp. 76–85 (2016)
26. Vuong, B.Q., He, Y., Hui, S.C.: Towards a web-based progressive handwriting recognition environment for mathematical problem solving. Expert Systems with Applications **37**(1), 886–893 (2010)
27. Wang, C., Zhang, H., Yang, L., Liu, S., Cao, X.: Deep people counting in extremely dense crowds. In: Proc. of ACM Multimedia. pp. 1299–1302 (2015)
28. Wang, J., Du, J., Zhang, J., Wang, Z.R.: Multi-modal attention network for handwritten mathematical expression recognition. In: Proc. of International Conference on Document Analysis and Recognition. pp. 1181–1186 (2019)
29. Winkler, H.J.: Hmm-based handwritten symbol recognition using on-line and off-line features. In: IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings. vol. 6, pp. 3438–3441 (1996)
30. Wu, J.W., Yin, F., Zhang, Y.M., Zhang, X.Y., Liu, C.L.: Image-to-markup generation via paired adversarial learning. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 18–34 (2018)
31. Wu, J.W., Yin, F., Zhang, Y.M., Zhang, X.Y., Liu, C.L.: Handwritten mathematical expression recognition via paired adversarial learning. International Journal of Computer Vision **128**(10), 2386–2401 (2020)



32. Xie, Z., Huang, Y., Zhu, Y., Jin, L., Liu, Y., Xie, L.: Aggregation cross-entropy for sequence recognition. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition. pp. 6538–6547 (2019)
33. Xu, C., Liang, D., Xu, Y., Bai, S., Zhan, W., Bai, X., Tomizuka, M.: Autoscale: Learning to scale for crowd counting. International Journal of Computer Vision pp. 1–30 (2022)
34. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: Proc. of Intl. Conf. on Machine Learning. pp. 2048–2057 (2015)
35. Yan, Z., Yuan, Y., Zuo, W., Tan, X., Wang, Y., Wen, S., Ding, E.: Perspective-guided convolution networks for crowd counting. In: Proc. of IEEE Intl. Conf. on Computer Vision (2019)
36. Yang, Y., Li, G., Wu, Z., Su, L., Huang, Q., Sebe, N.: Weakly-supervised crowd counting learns from sorting rather than locations. In: Proc. of European Conference on Computer Vision (2020)
37. Yuan, Y., Liu, X., Dikubab, W., Liu, H., Ji, Z., Wu, Z., Bai, X.: Syntax-aware network for handwritten mathematical expression recognition. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition. pp. 4553–4562 (2022)
38. Zeiler, M.D.: Adadelata: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
39. Zhang, A., Yue, L., Shen, J., Zhu, F., Zhen, X., Cao, X., Shao, L.: Attentional neural fields for crowd counting. In: Proc. of IEEE Intl. Conf. on Computer Vision (2019)
40. Zhang, J., Du, J., Dai, L.: Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In: Proc. of Intl. Conf. on Pattern Recognition. pp. 2245–2250 (2018)
41. Zhang, J., Du, J., Dai, L.: Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. IEEE Transactions on Multimedia **21**(1), 221–233 (2018)
42. Zhang, J., Du, J., Yang, Y., Song, Y.Z., Wei, S., Dai, L.: A tree-structured decoder for image-to-markup generation. In: Proc. of Intl. Conf. on Machine Learning. pp. 11076–11085 (2020)
43. Zhang, J., Du, J., Zhang, S., Liu, D., Hu, Y., Hu, J., Wei, S., Dai, L.: Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. Pattern Recognition **71**, 196–206 (2017)
44. Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y.: Single-image crowd counting via multi-column convolutional neural network. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition (2016)
45. Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of freebies for training object detection neural networks. arXiv preprint arXiv:1902.04103 (2019)
46. Zhao, W., Gao, L., Yan, Z., Peng, S., Du, L., Zhang, Z.: Handwritten mathematical expression recognition with bidirectionally trained transformer. In: Proc. of International Conference on Document Analysis and Recognition. pp. 570–584 (2021)