Mathematical Expression Correction for Output constraints

Sunghyun Kim, Mooho Song, Seoyoung Lim, Jeongsik Pyo
Seoul National University, Seoul
{ksho719, anmh9161, lsy0174, jungsik93}@snu.ac.kr

Abstract

Various approaches have been proposed to address Optical Character Recognition (OCR) tasks using advanced deep learning models. However, Handwritten Mathematical Expression Recognition (HMER), which falls under the umbrella of OCR tasks, presents unique challenges due to the need to comprehend mathematical syntax structures and accurately recognize handwritten symbols. In this paper, we propose a novel method called Mathematical Expression Correction for Output Constraints (MECO) for HMER tasks. This proposed module injected structural constraints of mathematical expressions to Syntax-Aware Network (SAN), Counting-Aware Network (CAN) and Coverage information in the transformer decoder (CoMER) baseline models to improve the accuracy of the model. The constraints are defined as relationships between grammatical components of the mathematical expressions, particularly focusing on the rules governing brackets. The MECO module optimizes the Lagrangian of the constrained problem aiming that all constraints are satisfied while minimizing the loss function. We compare the performance of the baseline models with MECO module with baseline models SAN, CAN and CoMER on the CROHME 2014, 2016, 2019 dataset. The integration of the MECO module enhances the performance of the baseline models, as evidenced by a reduction in both the number of bracket constraint violations and the total counts of violations. Furthermore, the MECO integrated models yields superior expression recognition rates across the most of the datasets in comparison to other baseline models.

1. Introduction

Handwritten Mathematical Expression Recognition (HMER) is a task of digitizing handwritten mathematical formulas. HMER is essential for applications in document analysis including office automation, assistance for disabled people, and educational purposes. With the introduction of advanced deep learning models to solve text recognition problems, several methods that handle Optical Character Recognition (OCR) tasks were proposed. However, HMER, one of the OCR tasks, is a demanding task that requires both understanding of mathematical syntax structures and the recognition of handwritten symbols. Unlike a list of words in a text, mathematical expression contains complexities that may require human's intuitions to comprehend them. As shown below Figure 1, HMER should be able to distinguish between denominators and numerators in division, as well as superscripts and subscripts in symbols.

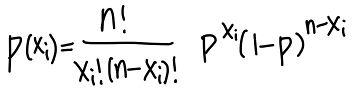
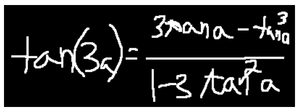
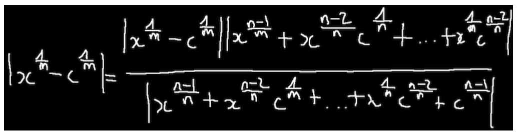


Figure 1. An example image used in HMER

Previous studies on HMER concentrated on the encoder-decoder image-to-sequence architectures [11, 14] with attention mechanism. These methods elevated the performance on the HMER tasks. Nevertheless, in the case of long markup sequences and crabbed handwritten expressions as shown below Figure 2, they could not successfully predict the sequences. The recent works on HMER called Counting-Aware Network (CAN) [6], Syntax-Aware Network (SAN) [9] and Coverage information in the Transformer Decoder (CoMER) [16] have been proposed to alleviate these problems.



(a) A crabbed handwriting example



(b) A long markup sequence example

Figure 2. Examples of challenging cases on HMER

CAN introduces a weakly-supervised counting module for symbol counting to promote the accuracy of attention results. SAN converts the markup sequence into a parsing tree and utilized a tree traverse process to predict the sequence effectively. CoMER employs an Attention Refinement Module to solve lack of coverage. For models in which data augumentaton is not utilized, they outperformed existing State-Of-The-Art (SOTA) models.

This paper aims to improve the accuracy of current SOTA models by introducing our method, Mathematical Expression Correction for Output Constraints (MECO). CROHME dataset [7] is used to train the baseline models. The monochrome input CROHME images, which have black background and white handwritten expressions, are fed to the encoder which extracts features from the images using convolution networks. The encoded vectors are transformed into mathematical sequences by applying attention modules. The models output the predicted sequence of mathematical expressions. Given the hard output constraints, we present MECO, a module for constraint injections, which forms a constrained optimization problem with Lagrangian variables [8]. We concentrated on injecting constraints regarding the balance of brackets which are common structural problems presented in the mathematical expressions. Initially, the baseline models with MECO module only train the baseline model for several epochs, while applying the gradient descent methods for baseline model's loss function. Then, the model starts to optimize baseline model's parameters and Lagrangian dual variables simultaneously, applying the gradient ascent methods for dual variables. Our method achieves successful injections of bracket constraints and hence outperformed the baseline models. This indicates that applying MECO module on models for HMER can surpass the existing models. In summary, our main contributions of this paper include the following :

1. We introduce a novel algorithm that incorporates constraint injection during the training phase for Handwritten Mathematical Expression Recognition (HMER). To the best of our knowledge, this is the first endeavor in this field.

2. We empirically show that the injection of a simple constraint, specifically regarding the existence of both opening and closing brackets, enhances the performance of the HMER models.

2. Related work

2.1. Handwritten Mathematics Expression Recognition

Conventionally, the Attention-based Encoder-Decoder structure has been extensively utilized for the Handwritten Mathematical Expression Recognition (HMER) tasks [10–12]. However, the performance of this method is limited by issues regarding the overlapping of current and past attention information maps.

To address these issues, models such as the Attention Aggregation Module (AAM) and the Bi-directional Mutual Learning Module (BML) have been proposed. These models employ attention-based encoder decoder architecture and are designed to facilitate one-to-one knowledge transfer during each training step, effectively synthesizing the complementary information from two inverse directions [2]. Another development was the introduction of a novel sequential relation decoder (SRD) to recognize the online mathematical expression (ME) in a tree structure, which more efficiently represents structural complexity. The tree is composed of a sub-tree sequence, where each sub-tree consists of a relation node and two symbol nodes [13]. The tree-based structures have been extended to incorporate directionality and stacking multiple layers to create deep Bidirectional Long-Short Term Memory (BLSTM). Despite achieving competitive results at the symbol level, even the advanced tree-based models demonstrated incomplete encapsulation of overall grammar structure both within the feature learning process of the neural networks, as well as in the decoder stage [15].

Specifically, the primary challenge of HMER task stems from the complexity of syntactic relationships rather than symbol recognition. To overcome these drawbacks, a tree-structured Syntax-Aware Network (SAN) model was proposed , which takes into account the relationships between grammatical elements during the grammar analysis process. Thus, compared to other tree-based models, SAN integrates syntax constraints into the feature learning stage of the deep neural networks. Further, SAN locates and recognizes the syntactically relevant components of MEs in the decoder stage, allowing it to capture complicated structural relationships. Indeed, this developed SAN model outperformed other tree-structured models [9].

On the other hand, the Counting-Aware Network (CAN) model applies a counting mechanism that aimed to improve the accuracy of the model by considering the occurrence and distribution of mathematical symbols within the expression. The decoder stage, similar to the SAN decoder, extracts the relevant syntax information from the MEs. In addition, the CAN model extracts features from the input images and learns the spatial distribution of symbols to improve the model's ability to capture context [6].

Finally, CoMER [16] introduces the coverage mechanism into the transformer decoder with Attention Refinement Module (ARM) to refine the attention weight in the transformer decoder, which effectively alleviates the lack of coverage problems without hurting its parallelism. Also, with the coverage mechanism, the authors refine attention weights and fully utilize the past alignment information generated from different layers in the stack transformer decoder. By applying these methodologies, CoMER outperforms other state-of-the-art RNN-based models or transformer-based models.

2.2. Constraint Injection for Neural Networks

A prevalent method for training a model is defining a loss function $L(w)$ with respect to a model parameter w, and solving the following optimization problem:

$$\underset{w}{\text{minimize}}\ L(w) \tag{1}$$

Sometimes, there might be demands to satisfy some constraints for output y of a model. Let $\{C_1, C_2, ..., C_K\}$ be constraints that we want our model to satisfy, $\mathcal{Y}_w$ be a range of model with parameter w, and denote a state that constraint C_k is satisfied as $\{f_k(y) \leq 0;\ \forall y \in \mathcal{Y}_w\}$ for $k = 1, 2, \cdots, K$. Since the model output y is determined by its parameter w, we denote $\{f_k(y) \leq 0;\ \forall y \in \mathcal{Y}_w\}$ as $\{f_k(w) \leq 0\}$ from now. Equipped with these constraints, the optimization problem becomes:

$$\begin{aligned} &\underset{w}{\text{minimize}}\ L(w) \\ &\text{subject to } f_k(w) \leq 0;\ \forall k = 1, 2, \cdots, K. \end{aligned} \tag{2}$$

There have been several types of research that is related to solving constrained optimization problems 2 with neural models by injecting hard constraints on output labels. Lee *et al.* [5] investigated Gradient-based Inference for output constraints. Their method injected constraints to the model at inference time with gradient-based parameter update, which reduces the effort for training a model again or resorting to expensive post-processing about discrete constraint space. Instead of using constrained optimzation form, they defined the minization problem using the regularizer term with origianl model's parameters. They successfully injected output constraints to neural models, even with performance enhancing for Semantic Role Labeling, Syntactic Parsing, and Synthetic Sequence Transduction. Nandwani *et al.* [8] injected constraints to neural models at training time. Their method converts the constrained optimization problem into a max-min problem using Lagrangian variables. While solving the penalized optimization problem, they also successfully injected output constraints into neural models even with performances enhancing in Semantic Role labeling, Named Entity Recognition, and Fine Grained Entity Typing.

3. Proposed Method

In this section, we introduce our constraints-injected model for Handwritten Mathematical Expression Recognition(HMER) task, **MECO**(**M**athematical **E**xpression **C**orrection for **O**utput constraints). We revise output hard-constraints that mathematical expression should satisfy. Given the baseline HMER model, we additionally build constraints injection training procedures. Our experiment uses the baseline as existing HMER models, and proposes a constraint injection training algorithm similar to that of Nandwani *et al.* [8]. The iteration Algorithm of **MECO** is described in 3.1, and the detailed mathematical form of constraints are described in 3.2.

3.1. Iteration Algorithm of MECO

Note that the corresponding Lagrangian of constrained problem 2 is:

$$\mathcal{L}(w, \mathcal{A}) = L(w) + \sum_{k=1}^{K} \lambda_k f_k(w) \tag{3}$$

, and solving 2 is equivalent to solving the following mini-max problem:

$$\min_{w} \max_{\mathcal{A}} \mathcal{L}(w, \mathcal{A}) \tag{4}$$

, where $\mathcal{A} = \{\lambda_k\}_{k=1}^{K}$ is set of non-negative Lagrangian multipliers. In Lagrangian 3, we set $L(w)$ as a loss function of baseline model, which can be directly obtained by baseline model. We describe constraints $\{f_k\}_{k=1}^{K}$ detail in section 3.2.

We convert the mini-max problem 4 into a max-min problem using the Lagrangian relaxation technique [1]) as follows:

$$\max_{\mathcal{A}} \min_{w} \mathcal{L}(w, \mathcal{A}) \tag{5}$$

, and the iteration process for solving max-min problem 5 uses following gradient update formulas:

$$W \leftarrow W - \alpha_p \nabla_W \mathcal{L} \tag{6}$$
$$\lambda_k \leftarrow \lambda_k + \alpha_d \nabla_{\lambda_k} \mathcal{L}. \tag{7}$$

The way to computationally solve the max-min problem 5 we used is shown in Algorithm 1. First, we initialize all of dual variables $\{\lambda_k\}_{k=1}^{K}$ as 0. Then, train only the baseline model until the warm-up epoch. After the warm-up epoch, we train dual variables $\mathcal{A}$ together while reducing the frequency of updating dual variables with gradient norm clipping of maximum gradient 100.

Note that, if we set f_i to be differentiable, then the Lagrangian 3 is also obviously differentiable, and hence the Lagrangian 3 is L-smooth function with $L = 100$ (100 is came from gradient clipping). Using the result of [4] and [3], we obtain that the convergence of Algorithm 1 guaranteed. The Algorithm 1 can be interpreted as us using two optimizers: one for the baseline model parameters w, and the other for the dual variable $\mathcal{A}$. For convenience, we denote the optimizer for baseline model as the 'primal optimizer', and the optimizer for the dual variable as the 'dual optimizer' from this point forward.

Algorithm 1 MECO's Algorithm

```
1:  Inputs: baseline model loss $L$, baseline model parameter $W$, constraints $f_1, f_2, \cdots, f_K$, corresponding dual variables $\mathcal{A} = \{\lambda_k\}_{k=1}^{K}$, warm-up epoch $w$, primal optimizer learning rate $\alpha_p$, dual optimizer learning rate $\alpha_d$, number of epoch $E$, constraint injection step $d$.
2:  initialize: $t, l, \lambda_1, \lambda_2 \cdots, \lambda_K \leftarrow 0$
3:  initialize: optimize-both $\leftarrow false$
4:  for each epoch $e = 1, 2, \cdots, E$ do
5:      if $l == t$ then
6:          $l \leftarrow l + d$
7:          $t \leftarrow 0$
8:          optimize-both $\leftarrow true$
9:      end if
10:     if $e \leq w$ then
11:         optimize-both $\leftarrow false$
12:     end if
13:     if optimizer-both$==$ $true$ then
14:         for $k = 1, 2, \cdots, K$ do
15:             $\lambda_k \leftarrow \lambda_k + \alpha_d \nabla_{\lambda_k} \mathcal{L}$        ▷ Gradient Ascent
16:             $\alpha_d \leftarrow \alpha_d^0 \frac{1}{1+d\beta}$
17:         end for
18:     end if
19:     $W \leftarrow W - \alpha_p \nabla_W \mathcal{L}$        ▷ Gradient Descent
20:     if $e > w$ then
21:         $t \leftarrow t + 1$
22:     end if
23:     optimizer-both$\leftarrow$ $false$
24: end for
```

3.2. Constraints

Balanced Brackets One of the fundamental constraints that a mathematical expression must satisfy is ensuring the proper balance of opening and closing brackets. Here, we denote bracket as three forms of brackets, '(', ')', '{', '}', '[', ']'. Note that, if there is an opening bracket(*e.g.*'('), there must be a closing bracket after it(*i.e.*')'). Let b_o, b_c be opening and closing brackets respectively. Assume that a mathematical expression sequence has a length of N, and an opening bracket b_o appears at the n-th sequence($n \in \{1, 2, \cdots, N\}$). We define a function g as follows:

$$g(w) = \begin{cases} \max\limits_{n+1 \leq i \leq N} P_w(y_i = b_c) & \text{for } 1 \leq n < N \\ 0 & \text{for } n = N \end{cases} \tag{8}$$

, where y_i is a i-th model output sequence. The constraint function $f(w)$ is defined by $f(w) = 1 - g(w)$. Note that $f(w) \in [0, 1]$ must hold by its definition. If there is no constraint violation, then $f(w) = 0$ holds. Otherwise, $f(w)$ is strictly greater than 0. Considering that the definition of f, $f(w) = 0$ is equivalent to $f(w) \leq 0$, which is a constraint part of the constrained optimization problems 2. For three types of brackets(parenthesis, brace, bracket), three constraint functions f_1, f_2, f_3 are defined in this way. If one b_o appears multiple times in a sequence, we consider n as the first location that b_o appears.

4. Experiments

In this section, we 1) provide concrete experiment settings, 2) introduce the datasets, and 3) compare results with previous models.

4.1. Experiment Settings

The baseline models and the models with MECO modules are implemented in Python, Pytorch Deep Learning Framework. Specifically, in the experimentation phase, both our CAN and CAN-MECO models, as well as SAN and SAN-MECO models, were trained on an NVIDIA RTX 3090 GPU with 24GB memory. All models were developed using PyTorch and deployed a batch size of 6. The learning rate was scheduled to warm up from 0 to 1 by the end of the first epoch and subsequently decayed in a cosine annealing manner. Adadelta, with a learning rate of 1, acted as the primary optimizer, while mini-batch Stochastic Gradient Ascent (SGA) was employed as the secondary optimizer with a learning rate of 0.1. Specifically, CAN and CAN-MECO models were trained using Python 3.8 over 240 epochs, whereas SAN and SAN-MECO models were implemented in Python 3.7.16 and trained for 200 epochs on the CROHME dataset. On the other hand, for the CoMER and CoMER-MECO models, we adopt a bidirectional training strategy akin to the one employed by a previous model(BTTR) in the HMER task. The PyTorch framework is utilized for implementation and deployed a batch size of 8. The optimization is carried out using Stochastic Gradient Descent (SGD) with a momentum of 0.9 and weight decay parameter set at 1e-4. The initial learning rate is configured at 0.08. For data augmentation, scale augmentation techniques are applied to the input images by uniformly sampling scaling factors within the range [0.7, 1.4].The experiments are executed on a computing cluster furnished with two NVIDIA RTX 3080 GPUs, each having 24GB of memory 1.

4.2. Datasets

The CROHME Dataset [7] is the most widely-used public dataset for HMER studies, which is the abbreviation of Competition on Recognition of Online Handwritten Mathematical Expression. CROHME contains online handwritten mathematical expressions in three editions: CROHME 2014, CROHME 2016, and CROHME 2019. The number of symbol classes is 111, including "sos" and "eos". The expressions were written by different people, including students and professionals, in various contexts, such as in classrooms or during lectures. The dataset also

Violation No.	Parenthesis	Brace	Bracket	Total
CoMER	35	33	6	74
CoMER-MECO(ours)	28	26	5	59
SAN	50	0	10	60
SAN-MECO(ours)	48	0	9	57
CAN	56	9	9	74
CAN-MECO(ours)	39	11	9	59

Violation Rate	Parenthesis	Brace	Bracket
CoMER	9.26	8.73	1.59
CoMER-MECO(ours)	7.84	7.28	1.40
SAN	9.6	0.0	1.9
SAN-MECO(ours)	9.8	0.0	1.8
CAN	12.6	2.0	2.0
CAN-MECO(ours)	8.9	2.5	2.0

Table 1. Constraints violation comparison for three types of brackets(parenthesis, brace, bracket) between baseline models and models including MECO module on the CROHME 2014 test set. **Left** represents the number of violated test set examples regarding number of opening and closing brackets. **Right** represents the violation rate(%) about numbers of opening, closing brackets. The denominator is a total number of mispredicted examples. A lower value is better for both measurements.

Method	CROHME 2014			CROHME 2016			CROHME 2019		
	ExpRate	≤ 1	≤ 2	ExpRate	≤ 1	≤ 2	ExpRate	≤ 1	≤ 2
CoMER [16]	61.73	70.14	83.67	60.24	69.14	78.52	61.13	69.33	80.76
CoMER-MECO(ours)	63.87	69.76	83.43	62.76	70.83	78.41	63.77	69.12	81.03
SAN [9]	47.66	65.01	73.33	45.07	61.81	71.58	45.29	63.14	72.48
SAN-MECO(ours)	50.71	66.53	71.81	48.04	63.34	70.01	44.37	60.05	69.22
CAN [6]	54.97	72.61	80.32	54.05	71.90	80.02	54.56	71.61	79.00
CAN-MECO(ours)	56.19	71.70	80.12	55.07	72.81	79.61	52.88	70.38	78.90

Table 2. This comparison presents the Expression Recognition Rate (ExpRate) and the tolerances of 1 and 2 score between baseline models and models with the MECO module on the CROHME 2014, 2016, and 2019 test datasets. Higher values indicate better performance.

includes ground-truth annotations at the symbol and expression level, which allows for the evaluation of recognition accuracy using standard metrics, such as character error rate and token accuracy.

We use 8,835 expressions with 111 symbols from CROHME dataset for training, and evaluate our model on CROHME 2014 test set, CROHME 2016 test set and CROHME 2019 test set, which consists of 986, 1147 and 1199 expressions, respectively. Finally, unlike many previous studies, we conducted experiments without performing data augmentation for a fair comparison.

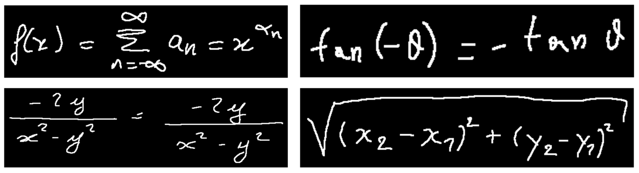


Figure 3. Some example images from the CROHME dataset

4.3. Result Comparison

Constraints Violation We define that the constraint violations have occurred when the numbers of b_o and b_c are different in a prediction sequence. Table 1 shows our constraints' violation rate and the number of constraints' violations occurred. The constraint violation rate is calculated by:

$$\text{Violation Rate} = \frac{\text{number of constraint violated examples}}{\text{number of mispredicted examples}}.$$

The results in Table 1 indicates that the MECO module successfully decreases both the number and rate of constraint violations. The CAN-MECO model reduced the total number of bracket constraint violations by 15 compared to the original CAN model. While the violation about parenthesis significantly reduced, the violation about brace increased. The SAN model has an intrinsic rule, so it does not violate constraints about braces. We observed that SAN-MECO does not effectively reduce the constraint violation. The SAN-MECO model reduced the total number of bracket constraint violation by 3 compared to the original SAN model. In the case of the CoMER-MECO model, it reduces the total number of mispredicted examples by 21 compared to the original CoMER model(Table 2). Among these 21 cases, 15 cases, which is about 70% of mispredicted examples, were bracket constraint violations. This implies that the balanced brackets constraints we injected genuinely improves the predictions of the baseline models, especially for parentheses, which is the most frequent case of bracket constraint violations. The number of parenthesis violations was reduced by 7, 2 and 15 for CoMER, SAN, and CAN respectively.

Expression Recognition Rate The Expression Recognition Rate(ExpRate) is a percentage of correctly predicted mathematical expressions in the model's predictions and is mostly used for the HMER task. At the symbol-level, we evaluate not only for exact match but also allow tolerances of 1 and 2 score, denoted by ≤ 1 and ≤ 2 in Table 2, respectively. Table 2 shows the ExpRate of CROHME 2014, 2016, 2019 dataset. As shown in Table 2, we enhanced the ExpRate performance of three models(without data augmentation), SAN, CAN and CoMER in most cases. SAN-MECO improved the ExpRate about 2% for the CROHME 14, 16 dataset compared to the original SAN model. CAN-MECO improved the ExpRate by more than 1% for CROHME 14, 16 dataset compared to original CAN model. However, the ExpRate of SAN-MECO and CAN-MECO decreased for the CROHME 19 dataset. Especially, the CoMER-MECO model outperformed the baseline model for every CROHME 14, 16, 19 dataset. Wth the CoMER-MECO model, ExpRate improved by more than 2% for every dataset. This implies that our constraints injection training procedure not only reduces constraint violations, but also maintains or even improves the baseline models' performance without affecting them in most cases.

5. Conclusion

In this paper, we proposed a new method called the Mathematical Expression Correction for Output constraints (MECO) module for handwritten mathematical expression recognition. The MECO module injects constraints that are intuitive to humans with regard to mathematical expressions. We chose to implement bracket constraints, which helps the models in balancing the number of opening and closing brackets. By applying the MECO module with the baseline models, namely SAN, CAN and CoMER on the CROHME dataset, our experiments validate two main conclusion: 1) The MECO module can successfully injects the opening and closing bracket constraints, especially with parenthesis, into the baseline models. 2) The MECO module consistently improves the accuracy of baseline models as reflected in the expression recognition rate. In future work, we are interested in seeking additional constraints to inject into the models to enhance their accuracy.

References

[1] DP Bertsekas. Nonlinear programming, athena scientific, belmont, massachusetts. *MR3444832*, 1999. 3

[2] Xiaohang Bian, Bo Qin, Xiaozhe Xin, Jianwu Li, Xuefeng Su, and Yanfeng Wang. Handwritten mathematical expression recognition via attention aggregation based bi-directional mutual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 113–121, 2022. 2

[3] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009. 3

[4] Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International conference on machine learning*, pages 4880–4889. PMLR, 2020. 3

[5] Jay Yoon Lee, Sanket Vaibhav Mehta, Michael Wick, Jean-Baptiste Tristan, and Jaime Carbonell. Gradient-based inference for networks with output constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4147–4154, 2019. 3

[6] Bohan Li, Ye Yuan, Dingkang Liang, Xiao Liu, Zhilong Ji, Jinfeng Bai, Wenyu Liu, and Xiang Bai. When counting meets hmer: Counting-aware network for handwritten mathematical expression recognition. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 197–214. Springer, 2022. 1, 2, 5

[7] Harold Mouchere, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 791–796. IEEE, 2014. 2, 4

[8] Yatin Nandwani, Abhishek Pathak, and Parag Singla. A primal dual formulation for deep learning with constraints. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 3

[9] Ye Yuan, Xiao Liu, Wondimu Dikubab, Hui Liu, Zhilong Ji, Zhongqin Wu, and Xiang Bai. Syntax-aware network for handwritten mathematical expression recognition, 2022. 1, 2, 5

[10] Jianshu Zhang, Jun Du, and Lirong Dai. A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 902–907. IEEE, 2017. 2

[11] Jianshu Zhang, Jun Du, and Lirong Dai. Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In *2018 24th international conference on pattern recognition (ICPR)*, pages 2245–2250. IEEE, 2018. 1, 2

[12] Jianshu Zhang, Jun Du, and Lirong Dai. 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. 2

[13] Jianshu Zhang, Jun Du, Yongxin Yang, Yi-Zhe Song, and Lirong Dai. Srd: a tree structure based decoder for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia*, 23:2471–2480, 2020. 2

[14] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206, 2017. 1

[15] Ting Zhang, Harold Mouchere, and Christian Viard-Gaudin. Tree-based blstm for mathematical expression recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 914–919. IEEE, 2017. 2

[16] Wenqi Zhao and Liangcai Gao. Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition, 2022. 1, 2, 5