Reproduction and Improvement of CoMER

Yuanjie Chen Zichen Zou

Abstract

CoMER is an effective transformer-based model for Handwritten Mathematical Expression Recognition. In this study, we reproduced the experimental results of CoMER and improved the model's performance through hyperparameter tuning and data augmentation. Additionally, we proposed an enhanced model architecture by augmenting DenseNet with multiple layers of various-sized convolutional kernels and integrating a self-attention module after image positional encoding. By combining these improvements, we achieved a comprehensive enhancement in the model's performance.

1. Introduction

Handwritten Mathematical Expression Recognition (HMER) focuses on converting handwritten mathematical expression images into corresponding LaTeX sequences. This technology has significant applications in online education, automatic scoring, and formula image searching. CoMER [12] is a transformer-based model innovately exploiting coverage information in the transformer decoder. The model introduces a new technique Attention Refinement Module (ARM) that dynamically refines attention weights using past alignment information, all without compromising parallelism. Self-coverage and Cross-coverage are used to fully utilize the past alignment information from the current and past layers. CoMER obtains significant improvement compared to other models and achieves expression recognition rates (ExpRate)s of 59.33% /59.81% /62.97% on the CROHME 2014 [9]/ 2016 [8]/ 2019 [7] datasets. In our experiment, we reproduce the CoMER model and improve the model performance by hyperparameter tuning, data augmentation. Then we propose an improved model architecture by enhancing the DenseNet with more Convolution-BatchNorm layers and adding a self-attention module after positional encoding of the image. We conducted ablation experiments with different parameters. Then we combined the above improvements to obtain a comprehensive improvement result.

2. CoMER

CoMER is named for **Co**verage information in the trans-for**MER** decoder.it is inspired by the coverage mechanism in RNN, it wants the transformer to allocate more attention to regions that have not yet been parsed. Specifically, it proposes a novel and general Attention Refinement Module (ARM) that dynamically refines the attention weights with past alignment information without hurting its parallelism. To fully use the past alignment information generated from different layers, it propose self-coverage and cross-coverage to utilize the past alignment information from the current and previous layer.Its structure is inherited by the baseline model BTTR [13].

We will introduce the basic structure of CoMER ,and the module improvements methods of CoMER in this section.

2.1. CoMER Structure

The CoMER model utilizes the coverage information as an attention refinement term in the transformer decoder without hurting its parallel decoding nature.Its structure is showed below.

2.2. Methodology

In this section, we will describe the architecture design of CoMER in detail. As illustrated in Fig.1, the model consists of four main modules: 1) CNN Encoder, which extracts features from 2D formula images. 2) Positional Encoding addresses position information for the transformer decoder. 3) Attention Refinement Module (ARM) is used to refine the attention weights with the past alignment information. 4) Self-coverage and cross-coverage utilize the past alignment information from the current and previous layer

2.2.1 CNN Encoder

In the encoder part, CoMER use DenseNet [3] to extract features in the 2D formula image, following the same setting with BTTR [13]. The core idea of DenseNet is to enhance the information flow between layers through concatenation operation in the feature dimension. Specifically, in the DenseNet block b, the output feature of lth layer can be computed by the output features $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{l-1} \in$

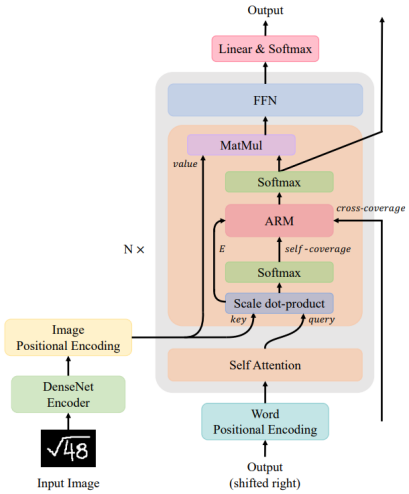

Figure 1. The overview architecture of proposed CoMER model. The attention weights generated by key and query are fed into a novel Attention Refinement Module (ARM). ARM utilizes the past alignment information generated from different layers through self-coverage and cross-coverage.

$\mathbb{R}^{h_b \times w_b \times d_b}$ from the previous 0th to $(l-1)$th layers:

$$\mathbf{X}_\ell = H_\ell([\mathbf{X}_0; \mathbf{X}_1; \ldots; \mathbf{X}_{\ell-1}]) \in \mathbb{R}^{h_b \times w_b \times d_b} \quad (1)$$

where $[\mathbf{X}_0; \mathbf{X}_1; \ldots; \mathbf{X}_{\ell-1}] \in \mathbb{R}^{h_b \times w_b \times (\ell d_b)}$ denotes the concatenation operation in the feature dimension, d_b denotes the feature dimension size of DenseNet block, and $H_\ell(\cdot)$ function is implemented by: a batch normalization [5] layer, a ReLU [2] activation function, and a 3×3 convolution layer.

In order to align DenseNet output feature with the model dimension size d_{model},the CoMER has a 1×1 convolution layer at the end of the encoder to obtain the output image feature $\mathbf{X}_e \in \mathbb{R}^{h_o \times w_o \times d_{\text{model}}}$.

2.2.2 Positional Encoding

Unlike the RNN decoders, which inherently consider the order of word tokens, the additional position information is necessary for the transformer decoder due to its permutation-invariant property. CoMER is consistent with BTTR [13], employing both image positional encoding and word positional encoding.

For word positional encoding, CoMER use the 1D positional encoding introduced in the vanilla transformer [11]. Given encoding dimension size d, position p, and the index i of feature dimension, the word positional encoding vector $\mathbf{P}^{\text{w}}_{p,d} \in \mathbb{R}^d$ can be represented as:

$$\mathbf{P}^{\text{w}}_{p,d}[2i] = \sin(p/10000^{2i/d}) \quad (2)$$

$$\mathbf{P}^{\text{w}}_{p,d}[2i+1] = \cos(p/10000^{2i/d}) \quad (3)$$

For image positional encoding, a 2D normalized positional encoding is used, which is the same as [1, 13]. Since it is not the absolute position but the relative position that matters, the position coordinates should be normalized first. Given a 2D coordinates tuple (x, y) and the encoding dimension size d, the image positional encoding tuple $\mathbf{P}^I_{x,y,d} \in \mathbb{R}^d$ is computed by the concatenation of 1D positional encoding (2,3) of two dimensions.

$$\bar{x} = \frac{x}{h_o}, \quad \bar{y} = \frac{y}{w_o} \quad (4)$$

$$\mathbf{P}^I_{x,y,d} = [\mathbf{P}^{\text{w}}_{\bar{x},d/2}; \mathbf{P}^{\text{w}}_{\bar{y},d/2}] \quad (5)$$

where h_o and w_o denote the shape of output image feature $\mathbf{X}_e \in \mathbb{R}^{h_o \times w_o \times d_{\text{model}}}$.

2.2.3 Attention Refinement Module

The CoMER build The **Attention Refinement Framework** ,which multiply the coverage matrix with $\mathbf{v}_a$ first and then add the result of LuongAttention [6].In this way,the space complexity will be greatly reduced to $O(TLh)$.We modify the computation of similarity vector e'_t as follows:

$$\begin{aligned} e'_t &= \tanh(\mathbf{H}_t\mathbf{W}_h + \mathbf{X}_f\mathbf{W}_x)\mathbf{v}_a + \mathbf{F}_t\mathbf{v}_a \\ &= \tanh(\underbrace{\mathbf{H}_t\mathbf{W}_h + \mathbf{X}_f\mathbf{W}_x}_{\text{attention}})\mathbf{v}_a + \underbrace{\mathbf{r}_t}_{\text{refinement}} \end{aligned} \quad (6)$$

where similarity vector e'_t can be divided into an attention term and a refinement $\mathbf{r}_t \in \mathbb{R}^L$ term. Notice that refinement term $\mathbf{r}_t$ could be directly produced by a coverage modeling function, avoiding intermediate representation with dimension d_{attn}.

To use this framework in the transformer, an *Attention Refinement Module (ARM)* is proposed shown in Fig. 2.

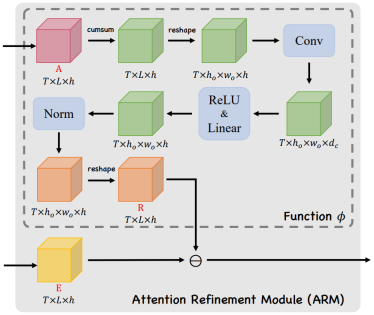

Figure 2. The overview of Attention Refinement Module (ARM).Given the generic attention weights A, the refinement term R was calculated using function$\phi(\cdot)$ at first.Then,the attention term $\mathbb{E}$ is refined by subtracting the refinement term R.

The scale dot-product matrix$\mathbb{E} \in \mathbb{R}^{T\times L\times h}$ can be used as the attention term, and the refinement term matrix $\mathbf{R}$ needs to be calculated from the attention weights $\mathbf{A}$. Note that a generic attention weights $\mathbf{A}$ is used here to provide past alignment information, and the specific choice of it will be discussed in Sect. 2.2.4

a function $\phi : \mathbb{R}^{T\times L\times h} \to \mathbb{R}^{T\times L\times h}$ takes the attention weights $\mathbf{A} \in \mathbb{R}^{T\times L\times h}$ as input and outputs the refinement matrix $\mathbf{R} \in \mathbb{R}^{T\times L\times h}$. With kernel size k_c, intermediate dimension d_c, $\mathbf{k} \in \mathbb{R}^{k_c\times k_c\times d_c}$, and the output image feature shape $L = h_o \times w_o$, function $\phi(\cdot)$ is defined as:

$$\mathbf{R} = \phi(\mathbf{A}) = \text{norm}\left(\max\left(0, \mathbf{K} * \tilde{\mathbf{C}} + \mathbf{b}_c\right)\mathbf{W}_c\right) \quad (7)$$

$$\tilde{\mathbf{C}} = \text{reshape}(\mathbf{C}) \in \mathbb{R}^{T\times h_o\times w_o\times h} \quad (8)$$

$$c_t = \sum_{k=1}^{t-1} a_k \in \mathbb{R}^{L\times h} \quad (9)$$

where a_t is the attention weights at step $t \in [0, T)$, $\mathbf{K} \in \mathbb{R}^{k_c\times k_c\times h_o\times d_c}$ denotes a convolution kernel, $*$ $\to$ convolution operation over the reshaped accumulated matrix $\tilde{\mathbf{C}} \in \mathbb{R}^{T\times h_o\times w_o\times h}$, $\mathbf{b}_c \in \mathbb{R}^{d_c}$ is a bias term, and $\mathbf{W}_c \in \mathbb{R}^{d_c\times h}$ is a linear projection matrix. Note that Eq. (9) can be efficiently computed by `cumsum(.)` function in modern deep learning frameworks [10].

function ϕ can extract local coverage features to detect the edge of parsed regions and identify the incoming unparsed regions. Finally, the attention term $\mathbb{E}$ is refined by subtracting the refinement term $\mathbf{R}$.

$$\text{ARM}(\mathbb{E}, \mathbf{A}) = \mathbb{E} - \mathbf{R} = \mathbb{E} - \phi(\mathbf{A}) \quad (10)$$

2.2.4 Coverage

Self-coverage and cross-coverage utilize the past alignment information from the current and previous layers, respectively. This helps in refining the attention mechanism in the transformer decoder.

Self-Coverage Self-coverage refers to using the alignment information generated by the current layer as input to the ARM. For the current layer j, we first calculate the attention weights $A^{(j)}$, and refine itself.

$$A^{(j)} = \text{softmax}(E^{(j)}) \in \mathbb{R}^{T\times L\times h} \quad (11)$$

$$\hat{E}^{(j)} = \text{ARM}(E^{(j)}, A^{(j)}) \quad (12)$$

$$\hat{A}^{(j)} = \text{softmax}(\hat{E}^{(j)}) \quad (13)$$

where $\hat{E}^{(j)}$ denotes the refined scale dot-product, and $\hat{A}^{(j)}$ denotes the refined attention weights at layer j.

Cross-Coverage A novel cross-coverage is proposed by exploiting the nature of the stacked decoder in the transformer. Cross-coverage uses the alignment information from the previous layer as input to the ARM of the current layer. For the current layer j, the refined attention weights $\hat{A}^{(j-1)}$ from the previous $(j-1)$ layer is used and the attention term of the current layer is refined.

$$\hat{E}^{(j)} = \text{ARM}(E^{(j)}, \hat{A}^{(j-1)}) \quad (14)$$

$$\hat{A}^{(j)} = \text{softmax}(\hat{E}^{(j)}) \quad (15)$$

Notice that $\hat{A}^{(j-1)} = A^{(j-1)}$ holds if the previous layer does not use the ARM.

Fusion-Coverage Combining the self-coverage and cross-coverage,a novel fusion-coverage method is proposed to fully use the past alignment information generated from different layers.

$$\hat{E}^{(j)} = \text{ARM}(E^{(j)}, [A^{(j)}; \hat{A}^{(j-1)}]) \quad (16)$$

$$\hat{A}^{(j)} = \text{softmax}(\hat{E}^{(j)}) \quad (17)$$

where $[A^{(j)}; \hat{A}^{(j-1)}] \in \mathbb{R}^{T\times L\times 2h}$ denotes the concatenation of attention weights from the current layer and refined attention weights from the previous layer.

3. Our approach

We adopt several methods to improve the performance of CoMER. One naive implementation is to fine-tune the hyperparameter such as the dropout rate. Considering that real-life handwritten formula images may have some degree of tilt or may not be entirely clear. Therefore, we have improved the data augmentation module by adding random rotation at certain angles and randomly adding Gaussian noise to enhance the model's generalization performance. Then we modify the encoder architecture by the self-attention technique and multilayers of convolution and batch-normalization.

3.1. Hyperparameter Tuning

Hyperparameter is crucial to the convergence and generalization ability of the models. Therefore hyperparameter tuning is an effective method to improve the performance of the mdoels whithout changing the dataset or the model structure. When we reproduced the experimental results of CoMER, we found that the training loss dropped rapidly to 0.01, but the validation loss remained at 0.3. There was significant overfitting. Therefore we adjust the hyperparameter dropout rate(from 0.3 to 0.4) to improve the model's generalization ability.

Another approach for hyperparameter tuning is to increase the dimension of the encoder to strengthen the ability of generalization. We increase the dimension of encoder from 256 to 512 to improve the performance of the model.

3.2. Data Augmentation

Considering that the images in the dataset may have some degree of tilt and low clarity due to the way they were captured. To improve the generalization ability, we added random rotation and Gaussian noise to the data augmentation process. In our experiment, We randomly rotated the images in the dataset by angles within five degrees and added Gaussian noise for blurring. The comparison of images before and after data augmentation is shown below.

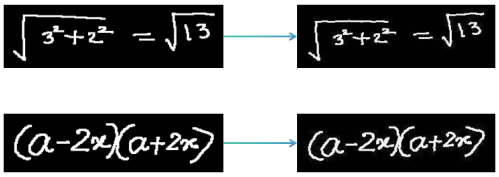

Figure 3. Examples of Data Augmentation

3.3. Enhanced DenseNet

In the CoMER model architecture, DenseNet [4] is used to extract the features in the 2D formula image.

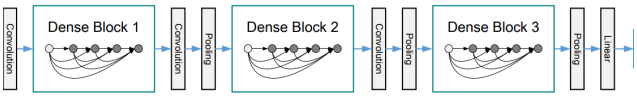

Figure 4. DenseNet Structure

One major challenge in HMER stems from the variability in handwriting styles and the high similarity between symbols (for example, 2 and z). So we add multiple layers with different sizes of convolutional kernels and Batch-Normalization to the original DenseNet to extract local features. In practice, we add 3×3 kernel size convolution layers and Batch-normalization layer to construct an enhanced DenseNet.

3.4. Self-Attention

Referring to the ViT model structure, we pass the position-encoded result through a multi-head self-attention layer before inputting it into the Transformer to Increase long-range dependencies and obtain alignment information of complex nested structure. The structure of the positional encoder after adding the self-attention layer is shown in figure 5.

4. Experiment

We trained the CoMER model on a server with four NVIDIA A800 GPUs and performed inference on the test set. Keeping the original configuration file unchanged, we

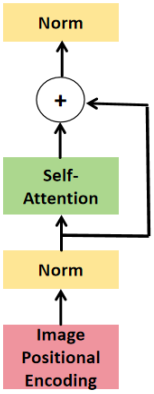

Figure 5. Self-Attention layer

reproduced the test results of the CoMER model. Due to time constraints, we stopped after training for 150 epochs. The ExpRate is shown in the table below.

	CROHME14	CROHME16	CROHME19
Original	59.33%	59.81%	62.97%
Reproduction	57.71%	58.5%	60.3%

Table 1. ExpRate Comparison between original results from the paper and the reproduction results

From the table we can know that our reproduction results are very close to the original results. To ensure fairness in comparison, we will compare the improved results after training 150 epochs of the following experiments with the reproduction results.

4.1. Hyperparameter Tuning

In the experiment, we increase the dropout rate from the orignial rate 0.3 to 0.4 and keep the other hyperparameter unchanged. The performance is shown in the table below.

dropout	CROHME14	CROHME16	CROHME19
0.3	**57.71%**	58.5%	60.3%
0.4	57.01%	**59.91%**	**62.31%**

Table 2. ExpRate Comparison between the reproduction result and the result after hyperparameter tuning

We can know from the table that we achieve better performance on CROHME16 and CROHME19 dataset by simply increasing the dropout rate.

Considering that models with a larger number of parameters generally have better generalization ability, we increased the model dimensions d_model from 256(original)

to 512. The performance on the test dataset is shown in the table below.

d_model	CROHME14	CROHME16	CROHME19
256	**57.71%**	**58.5%**	**60.3%**
512	53.14%	55.10%	57.88%

Table 3. ExpRate Comparison between the reproduction result and the result after hyperparameter tuning

We can know from the table that increasing the dimension of the model do not improve the performance. This might be because models with a larger number of parameters are harder to converge. Given our limited number of training epochs, it is difficult to achieve the expected results.

4.2. Data Augmentation

We applied improved data augmentation to the original dataset images by randomly flipping them within a 5-degree angle and adding Gaussian noise. We trained on the augmented dataset and compared the results. The performance is shown in the table below.

	CROHME14	CROHME16	CROHME19
Original	57.71%	58.5%	60.3%
Improved	**58.32%**	**60.33%**	**62.55%**

Table 4. ExpRate Comparison between the reproduction result and the result after Data Augmentation

We can find that the performance dominate on CROHME14,CROHME16 and CROHME19 dataset after applying improved data augmentation method.

4.3. Enhanced DenseNet

We add 3×3 kernel size convolution layer and BatchNormalization layer to enhance the ability of extracting features from the formula image. We trained the improved model for 150 epochs and the training loss changes as shown in the figure.

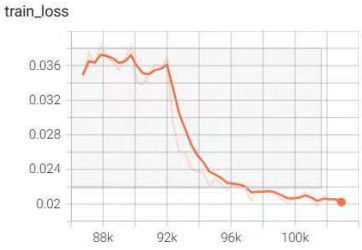

Figure 6. Training loss

We can see from the figure that the model training has reached convergence. Then we test the performance on the dataset. The performance is shown in the table below.

	CROHME14	CROHME16	CROHME19
Original	**57.71%**	58.5%	60.3%
Improved	56.70%	**59.98%**	**60.88%**

Table 5. ExpRate Comparison between the reproduction result and the result after Enhanced DenseNet

We can find that the performance improved on the CROHME16 and CROHME19 dataset after we enhance the DenseNet module.

4.4. Self-Attention

We add a Self-Attention layer after image positional encoding to better capture the alignment information. We trained the improved model for 150 epochs and the training loss changes as shown in the figure below.

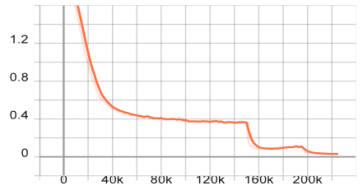

Figure 7. Training loss

The performance is shown in the table below.

	CROHME14	CROHME16	CROHME19
Original	**57.71%**	58.5%	60.3%
Improved	57.30%	**61.03%**	**60.80%**

Table 6. ExpRate Comparison between the reproduction result and the result after Self-Attention

We can find that the performance improved on CROHME16 and CROHME19 dataset after applying Self-Attention method.

4.5. Comprehensive Improvements

Considering that the previous improvements were made independently, we conducted experiments by combining the improvements mentioned above. The performance of comprehensive improvements is shown is the table below.

	CROHME14	CROHME16	CROHME19
D&E	56.19%	60.77%	60.22%
D&S	56.29%	56.49%	58.04%
E&S	53.14%	53.36%	57.63%
D&E&S	**57.31%**	**60.16%**	**60.55%**

Table 7. ExpRate of the Comprehensive Improvements, D is the abbreviation of Data Augmentation, E is the abbreviation of Enhanced Densenet, S is the abbreviation of Self-Attention

As can be seen from the above table, combining these improvements did not significantly enhance the model's performance. This is mainly because the change in the number of model parameters and the dataset resulted the difficulties in achieving convergence.

5. Ablation Study

To study the impact of hyperparameters on the improvement results, we conducted a parameter ablation experiment.

5.1. Data Augmentation

In the data augmentation section, we set the maximum rotation angle to 5 degrees and 10 degrees respectively for the experiments, and compared the improvement results. The result is shown in the table below.

Degree	CROHME14	CROHME16	CROHME19
5°	**58.32%**	**60.33%**	**62.55%**
10°	47.36%	51.23%	50.86%

Table 8. ExpRate Comparison between the result after rotating angle to 5 degrees and the result after rotating angle to 10 degrees

A 10-degree rotation of images in the training set results in excessive skew, which weakens the model's generalization ability.

5.2. Self-Attention

In the self-attention layer, we set the number of heads to 1 and 8 respectively for the experiments, and compared the impact of different parameters on the improvement results. The result is shown in the table below.

The 8-heads attention module has low performance because Multi-head attention introduces convergence difficulties.

head number	CROHME14	CROHME16	CROHME19
1	**57.30%**	**61.03%**	**60.80%**
8	56.39%	58.5%	57.8%

Table 9. ExpRate Comparison between the result of 1 head self-attention layer and the result of 8 heads self-attention layer

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. 2

[2] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. 2

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. 1

[4] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2016. 4

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 2

[6] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. 2

[7] Mahshad Mahdavi, Richard Zanibbi, Harold Mouchere, Christian Viard-Gaudin, and Utpal Garain. Icdar 2019 crohme + tfd: Competition on recognition of handwritten mathematical expressions and typeset formula detection. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1533–1538, 2019. 1

[8] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions. *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 607–612, 2016. 1

[9] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. 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, 2014. 1

[10] Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. X-linear attention networks for image captioning. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. 3

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 2

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

[13] Wenqi Zhao, Liangcai Gao, Zuoyu Yan, Shuai Peng, Lin Du, and Ziyin Zhang. Handwritten mathematical expression recognition with bidirectionally trained transformer. In Josep Lladós, Daniel Lopresti, and Seiichi Uchida, editors, *Document Analysis and Recognition – ICDAR 2021*, pages 570–584, Cham, 2021. Springer International Publishing. 1, 2

Contributions

Yuanjie Chen: Model Improvement

Zichen Zou: Environment configuration and Reproduction