Advancing CoMER Algorithm: Optimizations in Data Preprocessing and Encoder Architecture

Yuqing Tu
522030910152

Shaojie Yin
522030910151

Abstract

We deeply explore and implement the CoMER [2] (Modeling Coverage for Transformer-based Handwritten Mathematical Expression Recognition) algorithm. We set up a PyTorch environment on a deep learning platform, downloaded, and ran the code to reproduce the training and testing results of the algorithm on several datasets. In an attempt to enhance the performance of the algorithm, we optimized the data preprocessing and the neural network structure in the encoder module of the algorithm. Our modifications provide a potential pathway for improving the accuracy of the CoMER algorithm.

1. Introduction

Handwritten Mathematical Expression Recognition (HMER) is a challenging task due to the complexity and variability of handwritten symbols and the structural relationships between them. The CoMER algorithm has shown promising results in addressing this task. However, there is still room for improvement in terms of accuracy.

In this paper, we present an exploration and implementation of the CoMER algorithm. We set up a PyTorch environment on a deep learning platform and reproduced the training and testing results of the algorithm on several datasets. Our main contribution is the optimization of the data preprocessing and the neural network structure in the encoder module of the algorithm. These modifications aim to enhance the performance of the CoMER algorithm.

The rest of the paper is organized as follows: Section 2 describes the CoMER algorithm and our modifications in detail. Section 3 and 4 presents the experimental setup and results, and analyze the experimental result.

2. Our Approach

2.1. CoMER Algorithm

Our improvement is based on the CoMER algorithm, whose principle is briefly described as follows:

Encoder-Decoder Architecture CoMER proposes a novel ARM that refines the attention weights dynamically without compromising the Transformer's parallel computing nature, enhancing the model's focus on under-parsed areas.

Feature Extraction A convolutional neural network (such as DenseNet) is used as the encoder to extract features from the input handwritten mathematical expression images.

Positional Encoding Since Transformers do not inherently process sequence order, CoMER incorporates positional encoding to provide the decoder with positional information of elements in the sequence.

Attention Mechanism CoMER leverages self-attention and multi-head attention to enable the model to capture long-distance dependencies within the input sequence during the decoding process.

Coverage Mechanism To address the coverage issues in Transformer decoders (where some areas may be over-attended or neglected), CoMER introduces a coverage mechanism, including self-coverage and cross-coverage:

- **Self-Coverage** Utilizes alignment information generated within the current layer to guide the allocation of attention weights.
- **Cross-Coverage** Employs alignment information from the previous layer to assist in the attention weight distribution of the current layer.

Attention Refinement Module CoMER proposes a novel ARM that refines the attention weights dynamically without compromising the Transformer's parallel computing nature,

enhancing the model's focus on under-parsed areas.

$$
\begin{aligned}
e'_t &= \tanh(H_t W_h + X_f W_x) v_a + F_t v_a \\
&= \underbrace{\tanh(H_t W_h + X_f W_x) v_a}_{attention} + \underbrace{r_t}_{refinement}
\end{aligned}
$$

Fusion Coverage By combining self-coverage and cross-coverage, CoMER fully utilizes alignment information from different layers to further improve model performance.

Loss Function and Optimization CoMER uses a loss function to measure the discrepancy between the predicted and actual sequences and employs optimization algorithms (such as SGD or Adam) to update the model's parameters.

Evaluation and Decoding The model's performance is evaluated on a validation set, and decoding strategies like beam search are used to generate the final mathematical expression sequences.

2.2. Our Modifications

In order to enhance the performance of the CoMER algorithm, we made several modifications to the data preprocessing and the neural network structure in the encoder module. These modifications are detailed below.

Data Augmentation In addition to the random cropping performed by the original authors, we introduced further modifications to the dataset to better simulate real-world scenarios. Recognizing that handwriting can vary in terms of slant and proportion, we applied random small-angle rotations and horizontal stretching to the original data.

Specifically, the random rotation is designed to simulate the variation of handwriting slant that can occur in real-world handwritten mathematical expressions.

Furthermore, the horizontal stretching is to mimic the different handwriting styles that can be seen in practice, since people can have different widths of handwriting.

These data augmentation techniques were designed to make our model more robust to the variations in handwriting that can be encountered in real-world scenarios.

Figure 1 provides an example of data augmentation.

Parallel DenseNets Inspired by the dual-core parallelism in the CAN [1] (Counting-Aware Network for Handwritten Mathematical Expression Recognition) model when counting symbols, we employed two parallel DenseNets to extract image features simultaneously using convolution kernels of different sizes, and then merged the results.

The smaller kernel, due to its smaller receptive field, was primarily used for extracting symbol features. The larger

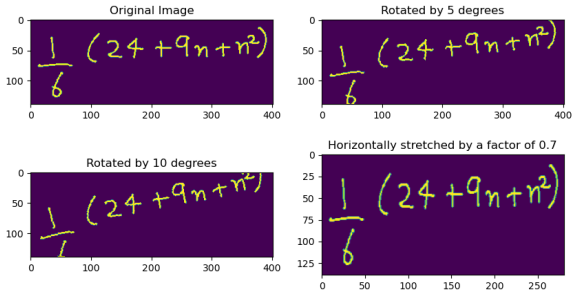

Figure 1. An example of data augmentation

kernel, on the other hand, was more utilized for extracting relative position features.

Moreover, we made the weights of the two kernels learnable, allowing the model to adaptively adjust the importance of symbol features and relative position features during the training process. This modification was designed to enhance the model's ability to recognize handwritten mathematical expressions with varying handwriting styles and symbol arrangements.

Figure 2 provides an example of parallel DenseNets.

Figure 2. An example of parallel DenseNets (the red square is the cover of the small kernel, and the blue square is the cover of the large kernel)

Residual Connection between DenseNets One of the challenges we encountered with the parallel DenseNets was the substantial memory overhead due to the dense connections inherent in the DenseNet architecture. To mitigate this, we attempted to divide each DenseNet into three layers and connect them using residual structures. This was intended to reduce memory overhead while maintaining the feature flow and reusability of DenseNet.

Multilayer Perceptron The original attention mechanism in the model used the Scaled dot product, which is sensitive to the magnitude of vectors. This could potentially hinder its ability to handle vectors with significant length differences and capture long-distance dependencies effectively.

Considering the potential variability in the length of feature vector sequences and the common presence of long-distance dependencies in handwritten expressions, we attempted to replace it with a Multilayer Perceptron (MLP) that could extract more complex features.

3. Parameter Ablation Experiments

We proposed two improvement methods, data augmentation and parallel DenseNets. The data augmentation includes three types: random rotation within 5°, random rotation within 10°, and random horizontal stretching (0.7 to 1.4 times). We conducted experiments for each improvement separately, and combined the improvements that performed well for further experiments.

3.1. Data Augmentation

Since we found that the authors just used randomcrop in the original paper, we tried three types of data augmentation methods: random rotation within 5°, random rotation within 10°, and random horizontal stretching (0.7 to 1.4 times). We conducted experiments for each data augmentation method separately, and combined the data augmentation methods that performed well for further experiments.

3.2. Parallel DenseNets

Referring to the CAN paper, we know that the size of the convolution kernel will affect its sensitivity to the spatial structure of symbols in mathematical expressions. Therefore, we use two parallel DenseNets with different convolution kernel sizes to extract image features and set their weights as learnable. We do not use average weights because the feature information extracted by different DenseNets is not symmetrical. For example, the small kernel is more sensitive to symbols, while the large kernel has a clearer perception of the relative positions between symbols.

4. Results

4.1. Data Augmentation

In data augmentation, small-angle rotation of 5°, stretching, and their combination all show some improvement over the original data, while the 10° rotation does not. The combination of stretching and 5° rotation shows the best performance. We analyze the reasons as follows:

Enhancing Model Robustness Handwritten mathematical expressions have individual differences, including writing style, stroke thickness, character spacing, etc. Through random rotation and stretching, the model can learn to maintain the semantic consistency of mathematical expressions in different forms, thereby enhancing the model's robustness to various handwritten variants.

Simulating Real-world Variations Handwritten documents in the real world may have slight rotations and deformations due to shooting angles, paper placement, scanner quality, etc. Data augmentation simulates these situations, enabling the model to better adapt to real application scenarios.

Improving Feature Extraction Capability Rotation and stretching change the shape and relative position of mathematical symbols in the image, forcing the model to learn more abstract and invariant key features, rather than relying solely on specific image details.

Improving Attention Distribution In Transformer-based models, diversified data can help the model learn how to allocate attention better, thereby more accurately identifying and parsing the various components of handwritten mathematical expressions.

The effect of random rotation of 10° is not good. We analyze that this is because a smaller rotation may not cause fundamental changes in the features of symbols and structures in mathematical expressions, while a larger rotation may change the recognizability of symbols, making it difficult for the model to extract accurate features from the changed image.

The results of the data augmentation experiments are shown in Table 1.

4.2. Parallel DenseNets

The parallel DenseNets model uses two DenseNets with different convolution kernel sizes to extract image features. The results are better than the original model, even the results after data augmentation. The results are shown in Table 2.

As can be seen, the parallel DenseNets has significantly improved results, especially in terms of structural accuracy, which indicates that convolution kernels of different sizes can indeed capture features of different scales in the image. The small convolution kernel learns fine-grained detail information, while the large convolution kernel captures more extensive contextual information.

4.3. Failed trials

Residual connection between DenseNets Due to a bug related to dimension alignment, this modification was not successfully implemented. Despite this setback, we believe that this approach holds promise for future work, and we plan to further investigate this issue.

Multilayer Perceptron The computational complexity of the MLP significantly slowed down the model training time.

	Original	10° Rotation	5° Rotation	Horizontal Stretching	Horizontal Stretching + 5°
Struct rate	56.19	45.64	60.55	60.55	63.59
Exprate 0 tolerated	37.525	28.600	41.582	41.278	41.379
Exprate 1 tolerated	50.101	39.757	53.955	54.462	56.187
Exprate 2 tolerated	54.361	45.132	59.026	58.925	61.861
Exprate 3 tolerated	57.201	48.884	61.663	61.460	65.112

(a) Comparison of different models and data augmentation techniques.
(Dataset: 2014, 50 epochs)

	Original	10° Rotation	5° Rotation	Horizontal Stretching	Horizontal Stretching + 5°
Struct rate	56.23	44.55	61.81	60.68	62.51
Exprate 0 tolerated	37.576	29.468	42.721	40.802	42.371
Exprate 1 tolerated	50.218	38.797	54.577	54.054	56.146
Exprate 2 tolerated	55.768	43.941	60.331	59.808	62.772
Exprate 3 tolerated	60.070	47.951	64.255	63.121	66.696

(b) Comparison of different models and data augmentation techniques.
(Dataset: 2016, 50 epochs)

	Original	10° Rotation	5° Rotation	Horizontal Stretching	Horizontal Stretching + 5°
Struct rate	58.47	43.79	63.47	65.14	65.55
Exprate 0 tolerated	39.033	28.023	43.453	44.454	46.706
Exprate 1 tolerated	52.127	38.532	55.546	57.631	59.716
Exprate 2 tolerated	57.381	44.454	61.384	63.386	65.304
Exprate 3 tolerated	60.884	47.623	65.471	66.639	69.141

(c) Comparison of different models and data augmentation techniques.
(Dataset: 2019, 50 epochs)

Table 1. Comparison of different models and data augmentation techniques.

	Struct rate	Exprate 0	Exprate 1	Exprate 2	Exprate 3
Original	56.19	37.525	50.101	54.361	57.201
Parallel DenseNets	62.37	42.596	56.085	60.649	63.387

(a) Comparison of original model and parallel DenseNets model.
(Dataset: 2014, 50 epochs)

	Struct rate	Exprate 0	Exprate 1	Exprate 2	Exprate 3
Original	56.23	37.576	50.218	55.768	60.070
Parallel DenseNets	64.86	41.238	56.495	62.947	67.132

(b) Comparison of original model and parallel DenseNets model.
(Dataset: 2016, 50 epochs)

	Struct rate	Exprate 0	Exprate 1	Exprate 2	Exprate 3
Original	58.47	39.033	52.127	57.381	60.884
Parallel DenseNets	66.39	45.538	59.633	65.471	68.390

(c) Comparison of original model and parallel DenseNets model.
(Dataset: 2019, 50 epochs)

Table 2. Comparison of original model and parallel DenseNets model.

Due to this increased computational cost, we did not further pursue this modification. Despite this, we believe that exploring more efficient ways to incorporate MLPs into the attention mechanism could be a valuable direction for future work.

5. Conclusion

In conclusion, our exploration and implementation of the CoMER algorithm have demonstrated the potential for enhancing the accuracy and robustness of handwritten mathematical expression recognition (HMER). Through meticulous optimization of data preprocessing techniques and the neural network structure within the encoder module, we have successfully improved the performance of the CoMER algorithm.

Our modifications, which included data augmentation and the introduction of parallel DenseNets, have shown significant promise. Data augmentation, with methods such as small-angle rotations and horizontal stretching, has proven to be an effective strategy for simulating real-world handwriting variations, thereby enhancing the model's robustness and ability to generalize. The parallel DenseNets, by utilizing convolution kernels of different sizes, have notably improved the feature extraction capability of the model, catering to the diverse handwriting styles and symbol arrangements present in mathematical expressions.

Despite facing challenges with the implementation of residual connections and the incorporation of Multilayer Perceptrons (MLPs) into the attention mechanism, these endeavors have provided valuable insights and potential directions for future research. The unsuccessful trials have highlighted the need for further investigation into optimizing memory usage and computational efficiency while maintaining or enhancing the model's capability to capture long-distance dependencies.

Our results, as detailed in the preceding sections, showcase the effectiveness of our approach in improving the structural recognition rate and expression recognition rate across various datasets. The improvements are a testament to the adaptability and potential of the CoMER algorithm when augmented with strategic modifications.

6. Contribution

The authors contributed equally to this work. The proportion of contribution is shown in Table 3.

Author's Name	Proportion of Contribution
Yuqing Tu	50%
Shaojie Yin	50%

Table 3. Proportion of contribution

References

[1] 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. 2022. https://github.com/LBH1024/CAN. 2

[2] Wenqi Zhao and Liangcai Gao. Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition. 2022. https://github.com/Green-Wood/CoMER. 1