



CoMER: Modeling Coverage for Transformer-based Handwritten Mathematical Expression Recognition

Wenqi Zhao¹  and Liangcai Gao¹ 

Wangxuan Institute of Computer Technology, Peking University, Beijing, China
wenqizhao@stu.pku.edu.cn, gaoliangcai@pku.edu.cn

Abstract. The Transformer-based encoder-decoder architecture has recently made significant advances in recognizing handwritten mathematical expressions. However, the transformer model still suffers from the lack of coverage problem, making its expression recognition rate (ExpRate) inferior to its RNN counterpart. Coverage information, which records the alignment information of the past steps, has proven effective in the RNN models. In this paper, we propose CoMER, a model that adopts the coverage information in the transformer decoder. Specifically, we propose a novel Attention Refinement Module (ARM) to refine the attention weights with past alignment information without hurting its parallelism. Furthermore, we take coverage information to the extreme by proposing self-coverage and cross-coverage, which utilize the past alignment information from the current and previous layers. Experiments show that CoMER improves the ExpRate by 0.61%/2.09%/1.59% compared to the current state-of-the-art model, and reaches 59.33%/59.81%/62.97% on the CROHME 2014/2016/2019 test sets.¹

Keywords: handwritten mathematical expression recognition · transformer · coverage · alignment · encoder-decoder model

1 Introduction

Handwritten mathematical expression recognition (HMER) aims to generate the corresponding \LaTeX sequence from a handwritten mathematical expression image. The recognition of handwritten mathematical expressions has led to many downstream applications, such as online education, automatic scoring, and formula image searching. During the COVID-19 pandemic, an increasing number of education institutions chose to use online platforms for teaching and examining. The recognition rate of handwritten mathematical expressions is crucial to improving both learning efficiency and teaching quality in online education scenarios.

Handwritten mathematical expression recognition is an image-to-text task with more challenges than traditional text recognition. Besides various writing

¹ Source code is available at <https://github.com/Green-Wood/CoMER>

styles, we also need to model the relationships between symbols and contexts [2]. In L^AT_EX, for example, the model needs to generate “^”, “_”, “{”, and “}” to describe the position and hierarchical relationship between symbols in a two-dimensional image. Researchers use the encoder-decoder architecture widely in the HMER task [9, 15, 26, 29, 32–35] because of its feature extraction in the encoder part and language modeling in the decoder part.

Transformer [28], a neural network architecture based solely on the attention mechanism, has gradually replaced RNN as the preferred model in natural language processing (NLP) [8]. Through the self-attention mechanism in the transformer, tokens in the same sequence establish direct one-to-one connections. Such an architecture allows the transformer to better model long-term dependency [3] between tokens. Currently, Transformer is attracting more and more attention in the computer vision [10] and multimodal [7, 17, 23] community.

Although transformer has become the standard de-facto in NLP, its performance in the HMER task was unsatisfactory compared with its RNN counterparts [9, 35]. We observe that the existing model using the transformer decoder still suffers from the lack of coverage problem [27, 34]. This problem manifests itself in two ways: over-parsing means that some parts of the image are unnecessarily parsed multiple times, while under-parsing means that some areas remain unparsed. RNN decoder uses coverage attention [9, 15, 26, 29, 32–34] to alleviate this problem. However, the current transformer decoder uses vanilla dot-product attention without the coverage mechanism, which is the key factor limiting its performance.

The computation of each step in the transformer is independent of each other, unlike RNN, where the computation of the current step depends on the previous step’s state. While this nature improves the parallelism in the transformer, it makes it difficult to use the coverage mechanism from previous works directly in the transformer decoder. To address the above issues, we propose a novel model for exploiting **C**overage information in the transfor**MER** decoder, named CoMER. Inspired by the coverage mechanism in RNN, we want the transformer to allocate more attention to regions that have not yet been parsed. Specifically, we propose 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, we propose self-coverage and cross-coverage to utilize the past alignment information from the current and previous layer, respectively. We further show that CoMER performs better than vanilla transformer decoder and RNN decoder in the HMER task. The main contributions of our work are summarized as follows:

- We propose a novel and general Attention Refinement Module (ARM) to refine the attention weight in the transformer decoder, which effectively alleviates the lack of coverage problem without hurting its parallelism.
- We propose self-coverage, cross-coverage, and fusion-coverage to fully use the past alignment information generated from different layers in the stack transformer decoder.

- Experiments show that CoMER outperforms existing state-of-the-art methods and achieves expression recognition rates (ExpRate)s of 59.33%/ 59.81%/ 62.97% on the CROHME 2014 [21]/2016 [22]/2019 [20] datasets.

2 Related Work

2.1 HMER Methods

The traditional approach usually divides the HMER task into two subtasks: symbol recognition and structure analysis [5]. Researchers represented the structural information of formulas through different predefined grammars, such as graph grammar [14], context-free grammar [1], and relational grammar [19]. These methods require researchers to develop hand-designed grammar rules, and their generalizability largely depends on the perfection of these grammar rules.

In recent years, encoder-decoder architectures have shown promising performance in various image-to-text tasks, such as scene text recognition [6] and image captioning [30]. In [34], a model called WAP was proposed to use encoder-decoder neural network for the first time to solve the HMER task and outperformed traditional grammar-based methods in the CROHME 2014 competition [21]. The WAP model uses a convolution neural network (CNN) encoder, a gated recurrent unit (GRU) decoder, and coverage attention to form the encoder-decoder architecture.

In terms of model architecture improvement, Zhang *et al.* [32] proposed DenseWAP, which uses a multi-scale DenseNet [12] encoder to improve the ability to handle multi-scale symbols. Ding *et al.* [9] then borrows the architecture design of the transformer to improve the RNN-based model performance by multi-head attention and stacked decoder.

In terms of data augmentation, Li *et al.* [15] proposed scale augmentation that scales the image randomly while maintaining the aspect ratio, which improves the generalization ability over multi-scale images. PAL-v2 [29] then uses printed mathematical expressions as additional data to help train the model.

In terms of training strategies, Truong *et al.* [26] proposed WS-WAP by introducing weakly supervised information about the presence or absence of symbols to the encoder. Besides, BTTR [35] was proposed to first use the transformer decoder for solving HMER task, and perform bidirectional language modeling with a single decoder.

2.2 Coverage Mechanism

The coverage mechanism was first proposed [27] to solve the over-translation and under-translation problems in the machine translation task.

All of the previous works in HMER [9, 15, 26, 27, 29, 32–34] used the coverage attention in RNN, where a coverage vector is introduced to indicate whether an image feature vector has been parsed or not, leading the model to put more attention on the unparsed regions. It is a step-wise refinement, where the decoder

needs to collect past alignment information for each step. For the RNN model, the decoder can naturally accumulate the attention weights in each step, but it is difficult for the transformer decoder which performs parallel decoding.

There is a work [25] that tried to introduce the coverage mechanism in transformer decoder. They directly used the coverage mechanism in RNN to the transformer, which greatly hurts its parallelism and training efficiency. Our CoMER model, on the other hand, utilizes the coverage information as an attention refinement term in the transformer decoder without hurting its parallel decoding nature.

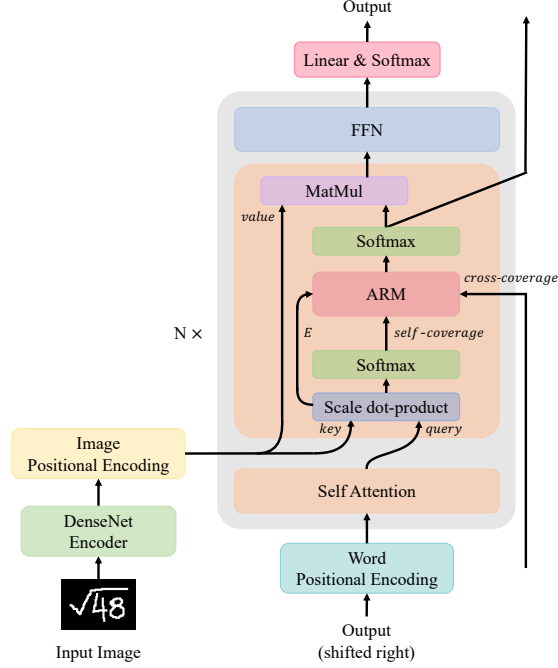


Fig. 1. The overview architecture of our proposed CoMER model. The attention weights generated by *key* and *query* are fed into a novel Attention Refinement Module (ARM). ARM utilizes the past alignment information generated from different layers through self-coverage and cross-coverage.

3 Methodology

In this section, we will first review the coverage attention in RNN and multi-head attention, then 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 that

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 layers.

3.1 Background

Coverage Attention in RNN Coverage attention has been widely used in RNN-based HMER models [9, 15, 26, 27, 29, 32–34]. The coverage vector provides information to the attention model about whether a region has been parsed or not. Let encoder produces flatten output image feature $\mathbf{X}_f \in \mathbb{R}^{L \times d_{\text{model}}}$ with the sequence length $L = h_o \times w_o$. At each step t , previous attention weights \mathbf{a}_k are accumulated as vector \mathbf{c}_t , and then transformed into coverage matrix \mathbf{F}_t .

$$\mathbf{c}_t = \sum_{k=1}^{t-1} \mathbf{a}_k \in \mathbb{R}^L \quad (1)$$

$$\mathbf{F}_t = \text{cov}(\mathbf{c}_t) \in \mathbb{R}^{L \times d_{\text{attn}}} \quad (2)$$

here $\text{cov}(\cdot)$ denotes a composite function of an 11×11 convolution layer and a linear layer.

In attention mechanism, we calculate a similarity score $e_{t,i}$ for every image feature at index $i \in [0, L)$. By taking advantage of broadcast operations in modern deep learning frameworks, such as PyTorch [24], we can compute similarity vector \mathbf{e}_t in parallel by broadcasting RNN hidden state $\mathbf{h}_t \in \mathbb{R}^{d_{\text{model}}}$ to $\mathbf{H}_t \in \mathbb{R}^{L \times d_{\text{model}}}$. The attention weights \mathbf{a}_t of current step t is obtained as follow:

$$\mathbf{e}_t = \tanh(\mathbf{H}_t \mathbf{W}_h + \mathbf{X}_f \mathbf{W}_x + \mathbf{F}_t) \mathbf{v}_a \quad (3)$$

$$\mathbf{a}_t = \text{softmax}(\mathbf{e}_t) \in \mathbb{R}^L \quad (4)$$

where $\mathbf{W}_h \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}}$, $\mathbf{W}_x \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}}$ are trainable parameters matrices, and $\mathbf{v}_a \in \mathbb{R}^{d_{\text{attn}}}$ is a trainable parameter vector.

Multi-Head Attention Multi-head attention is the most critical component of the transformer models [28]. With the model dimension size d_{model} , query sequence length T , and key sequence length L , we split the multi-head attention calculation for head Head_i into four parts: 1) project the query \mathbf{Q} , key \mathbf{K} , and value \mathbf{V} into a subspace; 2) calculate the scaled dot-product $\mathbf{E}_i \in \mathbb{R}^{T \times L}$; 3) compute the attention weights $\mathbf{A}_i \in \mathbb{R}^{T \times L}$ by the softmax function; 4) obtain head Head_i by multiplying the attention weights \mathbf{A}_i and value \mathbf{V}_i .

$$\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i = \mathbf{Q} \mathbf{W}_i^Q, \mathbf{K} \mathbf{W}_i^K, \mathbf{V} \mathbf{W}_i^V \quad (5)$$

$$\mathbf{E}_i = \frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d_k}} \in \mathbb{R}^{T \times L} \quad (6)$$

$$\mathbf{A}_i = \text{softmax}(\mathbf{E}_i) \in \mathbb{R}^{T \times L} \quad (7)$$

$$\mathbf{Head}_i = \mathbf{A}_i \mathbf{V}_i \in \mathbb{R}^{T \times d_v} \quad (8)$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ denote the trainable projection parameter matrices. Then all h heads are concatenated and projected with a trainable projection matrix $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ to obtain the final output:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathbf{Head}_1; \dots; \mathbf{Head}_h] \mathbf{W}^O \quad (9)$$

We follow this setting in CoMER and use the Attention Refinement Module (ARM) in Sect. 3.4 to refine the scale dot-product matrix \mathbf{E}_i in Eq. (6).

3.2 CNN Encoder

In the encoder part, we use DenseNet [12] to extract features in the 2D formula image, following the same setting with BTTR [35]. 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 l^{th} layer can be computed by the output features $\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_{l-1} \in \mathbb{R}^{h_b \times w_b \times d_b}$ from the previous 0^{th} to $(l-1)^{\text{th}}$ layers:

$$\mathbf{X}_\ell = H_\ell([\mathbf{X}_0; \mathbf{X}_1; \dots; \mathbf{X}_{\ell-1}]) \in \mathbb{R}^{h_b \times w_b \times d_b} \quad (10)$$

where $[\mathbf{X}_0; \mathbf{X}_1; \dots; \mathbf{X}_{\ell-1}] \in \mathbb{R}^{h_b \times w_b \times (ld_b)}$ 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 [13] layer, a ReLU [11] activation function, and a 3×3 convolution layer.

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

3.3 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. In CoMER, we are consistent with BTTR [35], employing both image positional encoding and word positional encoding.

For word positional encoding, we use the 1D positional encoding introduced in the vanilla transformer [28]. Given encoding dimension size d , position p , and the index i of feature dimension, the word positional encoding vector $\mathbf{p}_{p,d}^{\mathbf{W}} \in \mathbb{R}^d$ can be represented as:

$$\begin{aligned} \mathbf{p}_{p,d}^{\mathbf{W}}[2i] &= \sin(p/10000^{2i/d}) \\ \mathbf{p}_{p,d}^{\mathbf{W}}[2i+1] &= \cos(p/10000^{2i/d}) \end{aligned} \quad (11)$$

For image positional encoding, a 2D normalized positional encoding is used, which is the same as [4, 35]. 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 $\mathbf{p}_{x,y,d}^{\mathbf{I}} \in \mathbb{R}^d$ is computed by the concatenation of 1D positional encoding (11) of two dimensions.

$$\bar{x} = \frac{x}{h_o}, \quad \bar{y} = \frac{y}{w_o} \quad (12)$$

$$\mathbf{p}_{x,y,d}^{\mathbf{I}} = [\mathbf{p}_{\bar{x},d/2}^{\mathbf{W}}; \mathbf{p}_{\bar{y},d/2}^{\mathbf{W}}] \quad (13)$$

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

3.4 Attention Refinement Module

Although coverage attention has been widely used in the RNN decoder, it is difficult to use it directly in the transformer decoder due to transformer’s parallel decoding. The inability to model the coverage information directly in the transformer leads to its unsatisfactory performance in the HMER task. We will first introduce the difficulty of using coverage information in the transformer in this subsection, then propose a novel Attention Refinement Module (ARM) to solve it.

A naive solution is to use the multi-head attention weight \mathbf{A} in Eq. (7), accumulate it as \mathbf{C} , and then transform into a coverage matrix \mathbf{F} using $\text{cov}(\cdot)$ function in Eq. (2). However, this naive solution is unacceptable considering space complexity. Assume that multi-head attention weight $\mathbf{A} \in \mathbb{R}^{T \times L \times h}$, then $\text{cov}(\cdot)$ function will be applied at every time step and every image feature location, which will produce coverage matrix $\mathbf{F} \in \mathbb{R}^{T \times L \times h \times d_{\text{attn}}}$ with space complexity $O(TLhd)$.

We can see that the bottleneck comes from the $\tanh(\cdot)$ function in Eq. (3) where the coverage matrix needs to be summed with other feature vectors first, then multiplied by vector $\mathbf{v}_a \in \mathbb{R}^{d_{\text{attn}}}$. If we can multiply the coverage matrix with \mathbf{v}_a first and then add the result of LuongAttention [18], the space complexity will be greatly reduced to $O(TLh)$. So we modify Eq. (3) as follows:

$$\begin{aligned} \mathbf{e}'_t &= \tanh(\mathbf{H}_t \mathbf{W}_h + \mathbf{X}_f \mathbf{W}_x) \mathbf{v}_a + \mathbf{F}_t \mathbf{v}_a \\ &= \underbrace{\tanh(\mathbf{H}_t \mathbf{W}_h + \mathbf{X}_f \mathbf{W}_x) \mathbf{v}_a}_{\text{attention}} + \underbrace{\mathbf{r}_t}_{\text{refinement}} \end{aligned} \quad (14)$$

where similarity vector \mathbf{e}'_t can be divided into an attention term and a refinement $\mathbf{r}_t \in \mathbb{R}^L$ term. Notice that given accumulated vector \mathbf{c}_t in Eq. (1), refinement term \mathbf{r}_t could be directly produced by a coverage modeling function, avoiding intermediate representation with dimension d_{attn} . We name the process in Eq. (14) as the **Attention Refinement Framework**.

To use this framework in the transformer, we propose an **Attention Refinement Module (ARM)** shown in Fig. 2. The scale dot-product matrix

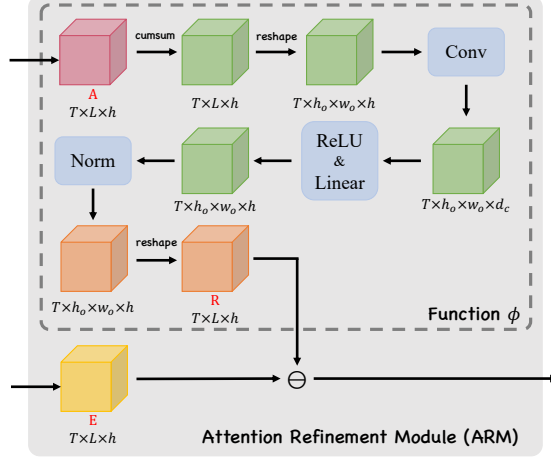


Fig. 2. The overview of Attention Refinement Module (ARM). Given the generic attention weights \mathbf{A} , we first calculate the refinement term \mathbf{R} using function $\phi(\cdot)$. Then, we refine the attention term \mathbf{E} by subtracting the refinement term \mathbf{R} .

$\mathbf{E} \in \mathbb{R}^{T \times L \times h}$ in Eq. (6) 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 we use a generic attention weights \mathbf{A} here to provide past alignment information, and the specific choice of it will be discussed in Sect. 3.5.

We define a function $\phi : \mathbb{R}^{T \times L \times h} \mapsto \mathbb{R}^{T \times L \times h}$ that 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 $d_c \ll h \times d_{\text{attn}}$, and the output image feature shape $L = h_o \times w_o$, function $\phi(\cdot)$ is defined as:

$$\mathbf{R} = \phi(\mathbf{A}) = \text{norm} \left(\max \left(0, \mathbf{K} * \tilde{\mathbf{C}} + \mathbf{b}_c \right) \mathbf{W}_c \right) \quad (15)$$

$$\tilde{\mathbf{C}} = \text{reshape}(\mathbf{C}) \in \mathbb{R}^{T \times h_o \times w_o \times h} \quad (16)$$

$$\mathbf{c}_t = \sum_{k=1}^{t-1} \mathbf{a}_k \in \mathbb{R}^{L \times h} \quad (17)$$

where \mathbf{a}_t is the attention weights at step $t \in [0, T)$, $\mathbf{K} \in \mathbb{R}^{k_c \times k_c \times h \times d_c}$ denotes a convolution kernel, $*$ denotes 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. (17) can be efficiently computed by $\text{cumsum}(\cdot)$ function in modern deep learning frameworks [24].

We consider that function ϕ can extract local coverage features to detect the edge of parsed regions and identify the incoming unparsed regions. Finally, we

refine the attention term \mathbf{E} by subtracting the refinement term \mathbf{R} .

$$\begin{aligned}\text{ARM}(\mathbf{E}, \mathbf{A}) &= \mathbf{E} - \mathbf{R} \\ &= \mathbf{E} - \phi(\mathbf{A})\end{aligned}\tag{18}$$

3.5 Coverage

In this section, we will discuss the specific choice of the generic attention weights \mathbf{A} in Eq. (15). We propose self-coverage and cross-coverage to utilize alignment information from different stages, introducing diverse past alignment information to the model.

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

$$\mathbf{A}^{(j)} = \text{softmax}(\mathbf{E}^{(j)}) \in \mathbb{R}^{T \times L \times h}\tag{19}$$

$$\hat{\mathbf{E}}^{(j)} = \text{ARM}(\mathbf{E}^{(j)}, \mathbf{A}^{(j)})\tag{20}$$

$$\hat{\mathbf{A}}^{(j)} = \text{softmax}(\hat{\mathbf{E}}^{(j)})\tag{21}$$

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

Cross-coverage We propose a novel cross-coverage 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 , we use the refined attention weights $\hat{\mathbf{A}}^{(j-1)}$ from the previous $(j-1)$ layer and refine the attention term of the current layer.

$$\hat{\mathbf{E}}^{(j)} = \text{ARM}(\mathbf{E}^{(j)}, \hat{\mathbf{A}}^{(j-1)})\tag{22}$$

$$\hat{\mathbf{A}}^{(j)} = \text{softmax}(\hat{\mathbf{E}}^{(j)})\tag{23}$$

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

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

$$\hat{\mathbf{E}}^{(j)} = \text{ARM}(\mathbf{E}^{(j)}, [\mathbf{A}^{(j)}; \hat{\mathbf{A}}^{(j-1)}])\tag{24}$$

$$\hat{\mathbf{A}}^{(j)} = \text{softmax}(\hat{\mathbf{E}}^{(j)})\tag{25}$$

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

4 Experiments

4.1 Implementation Details

In the encoder part, we employ the same DenseNet to extract features from the formula image. Three densenet blocks are used in the encoder, each containing $D = 16$ bottleneck layers. A transition layer is inserted between every two densenet blocks to reduce the spatial and channel size of the feature map by $\theta = 0.5$. The growth rate is set to $k = 24$, and the dropout rate is set to 0.2.

In the decoder part, for hyperparameters in the transformer decoder, we set the model dimension to $d_{\text{model}} = 256$, the number of heads to $h = 8$, and the feed-forward layer dimension size to $d_{ff} = 1024$. We use three stacked decoder layers and a 0.3 dropout rate. For the Attention Refinement Module that we proposed, we set the kernel size to $k_c = 5$, the intermediate dimension to $d_c = 32$. The normalization method we adopt in ARM is batch-normalization [13]. We use ARM starting with the second layer and share the same ARM between layers.

We use the same bidirectional training strategy as BTTR [35] to train CoMER with PyTorch [24] framework. We use SGD with a weight decay of 10^{-4} and momentum of 0.9. The learning rate is 0.08. We augment input images using scale augmentation [15] with uniformly sampled scaling factor $s \in [0.7, 1.4]$. All experiments are conducted on four NVIDIA 2080Ti GPUs with 4×11 GB memory.

In the inference phase, instead of the beam search, we perform the approximate joint search [16] that has been used in BTTR [35].

4.2 Datasets and Metrics

We use the Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) datasets [20–22] to conduct our experiments, which is currently the largest open dataset for HMER task. The training set contains a total of 8836 training samples, while the CROHME 2014/2016/2019 test set contains 986/1147/1199 test samples. The CROHME 2014 test set [21] is used as the validation set to select the best-performing model during the training process.

We use the evaluation tool officially provided by the CROHME 2019 [20] organizers to convert the predicted \LaTeX sequences into symLG format. Then, metrics are reported by utilizing the LgEval library [31]. We choose “ExpRate”, “ ≤ 1 error”, “ ≤ 2 error”, and “ ≤ 3 error” metrics to measure the performance of our proposed model. These metrics represent the expression recognition rate when we tolerate 0 to 3 symbol or structural errors.

4.3 Ablation Study

To verify the effectiveness of our proposed method, we performed ablation experiments on the CROHME 2014 test set [21]. In Table 1, the “Scale-aug” column indicates whether to adopt scale augmentation [15] for data augmentation. The

“Self-cov” column indicates whether self-coverage is used. The “Cross-cov” column suggests the use of cross-coverage.

First, since the original BTTR [35] did not use any data augmentation methods, we re-implement the BTTR model and achieve an ExpRate of 53.45%, which is similar to the original result in [35]. To compare BTTR as a baseline with our proposed CoMER, we also use scale augmentation to train BTTR and obtained an ExpRate of 55.17% for “BTTR (baseline)”.

The performance of CoMER using ARM and coverage mechanism has been significantly improved compared to BTTR. Comparing the last four rows in Table 1, we can observe that:

1. When CoMER uses self-coverage to refine the attention weights, the performance is improved by 2.34% compared to “BTTR (baseline)”. Experiment results validate the feasibility and effectiveness of using past alignment information in the transformer decoder.
2. Compared to self-coverage, using cross-coverage in CoMER can bring more performance gains owing to the more accurate alignment information from the previous layer.
3. “CoMER (Fusion)” obtains the best results by combining self-coverage and cross-coverage, outperforming the baseline model by 4.16%. This experiment results suggest that diverse alignment information generated from different layers helps ARM refine the current attention weights more accurately.

Table 1. Ablation study on the CROHME 2014 test set (in %). † denotes original reported results of BTTR [35]

Model	Scale-aug [15]	Self-cov	Cross-cov	ExpRate
BTTR [†] [35]	✗	✗	✗	53.96
BTTR	✗	✗	✗	53.45
BTTR (baseline)	✓	✗	✗	55.17 (+0.00)
CoMER (Self)	✓	✓	✗	57.51 (+2.34)
CoMER (Cross)	✓	✗	✓	58.11 (+2.94)
CoMER (Fusion)	✓	✓	✓	59.33 (+4.16)

4.4 Comparison with State-of-the-art Approaches

We compare the proposed CoMER with the previous state-of-the-art methods, as shown in Table 2. For the RNN-based models, we choose DenseWAP [32], DenseWAP-TD [33], WS-WAP [26], Li *et al.* [15], and Ding *et al.* [9] for comparison. For transformer-based models, we compare with BTTR [35] that uses a

vanilla transformer decoder. Note that the approaches proposed by Li *et al.* [15] and Ding *et al.* [9] also use the scale augmentation [15].

Compared with the RNN-based models that use coverage attention, CoMER outperforms the previous state-of-the-art model proposed by Ding *et al.* [9] on each CROHME test set. In the ExpRate metric, CoMER improves by an average of 1.43% compared to the previous best-performing RNN-based model.

Compared with the transformer-based model, our proposed CoMER equipped with ARM and fusion-coverage significantly improves the performance. Specifically, CoMER outperforms “BTTR (baseline)” in all metrics and averages 3.6% ahead of “BTTR (baseline)” in ExpRate.

Table 2. Performance comparison with previous state-of-the-art approaches on the CROHME 2014/2016/2019 test sets (in %).

Dataset	Model	ExpRate	≤ 1 error	≤ 2 error	≤ 3 error
CROHME 14	DenseWAP [32]	43.0	57.8	61.9	-
	DenseWAP-TD [33]	49.1	64.2	67.8	-
	WS-WAP [26]	53.65	-	-	-
	Li <i>et al.</i> [15]	56.59	69.07	75.25	78.60
	Ding <i>et al.</i> [9]	58.72	-	-	-
	BTTR [35]	53.96	66.02	70.28	-
	BTTR (baseline)	55.17	67.85	72.11	74.14
	CoMER	59.33	71.70	75.66	77.89
CROHME 16	DenseWAP [32]	40.1	54.3	57.8	-
	DenseWAP-TD [33]	48.5	62.3	65.3	-
	WS-WAP [26]	51.96	64.34	70.10	72.97
	Li <i>et al.</i> [15]	54.58	69.31	73.76	76.02
	Ding <i>et al.</i> [9]	57.72	70.01	76.37	78.90
	BTTR [35]	52.31	63.90	68.61	-
	BTTR (baseline)	56.58	68.88	74.19	76.90
	CoMER	59.81	74.37	80.30	82.56
CROHME 19	DenseWAP [32]	41.7	55.5	59.3	-
	DenseWAP-TD [33]	51.4	66.1	69.1	-
	Ding <i>et al.</i> [9]	61.38	75.15	80.23	82.65
	BTTR [35]	52.96	65.97	69.14	-
	BTTR (baseline)	59.55	72.23	76.06	78.40
	CoMER	62.97	77.40	81.40	83.07

4.5 Performance at Different Lengths

Intuitively, we assume that the recognition accuracy of long sequences is lower than that of short ones because of the lack of coverage problem [27, 34]. Thus, we consider the recognition accuracy of long sequences reflects the ability of a model to align the sequence and image. To verify that CoMER has better alignment

and thus alleviates the lack of coverage problem, we calculate the recognition accuracy at different lengths on the CROHME 2014 test set, shown in Fig. 3.

By comparing “BTTR (baseline)” with the CoMER family of models, we found that CoMER equipped with ARM has better performance when dealing with various lengths of sequences, especially with longer ones. The performance of “CoMER (Fusion)” can reach even $5\times$ than “BTTR (baseline)” when recognizing sequences longer than 50. This experiment results show that the ARM and coverage mechanisms can improve the alignment quality and mitigates the lack of coverage problem.

Comparing the performance between self-coverage and cross-coverage, we find that cross-coverage performs better when parsing short sequences. In contrast, self-coverage is better at recognizing long sequences. We suppose this is because cross-coverage accumulates misalignments generated by the previous layer, causing it to incorrectly refine the attention weights in the current layer. In comparison, self-coverage performs alignment and refinement in each layer independently. “CoMER (Fusion)” uses both self-coverage and cross-coverage to exploit diverse alignment information and far outperforms other models in recognizing sequences longer than 20.

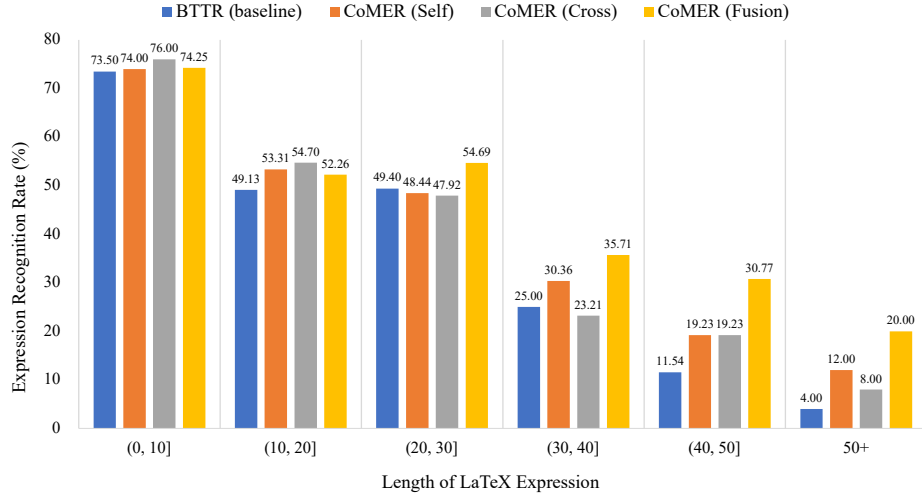


Fig. 3. Recognition accuracy (in %) for \LaTeX expressions of different lengths on the CROHME 2014 test set.

4.6 Refinement Term Visualization

As shown in Fig. 4, we visualize the refinement term \mathbf{R} in the recognition process. We find that parsed regions are darker, which indicates ARM will suppress

the attention weights in these parsed regions and encourage the model to focus on incoming unparsed regions. The visualization experiment shows that our proposed ARM can effectively alleviate the lack of coverage problem.

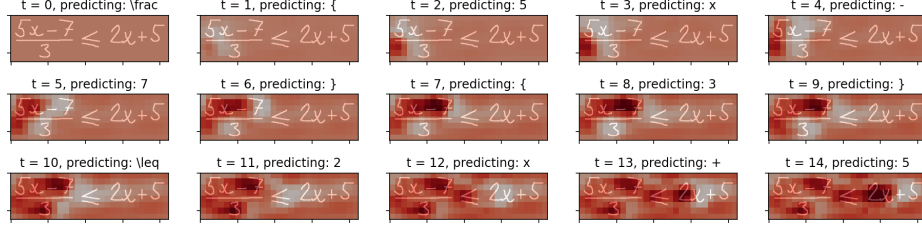


Fig. 4. The refinement term \mathbf{R} visualization in recognizing formula image. The darker the color, the larger the value.

5 Conclusion and Future Work

In this paper, inspired by the coverage attention in RNN, we propose CoMER to introduce the coverage mechanism into the transformer decoder. We have the following four main contributions: **(1)** Our proposed CoMER alleviates the lack of coverage problem and significantly improves the recognition accuracy of long \LaTeX expressions. **(2)** We propose a novel Attention Refinement Module (ARM) that makes it possible to perform attention refinement in the transformer without harming its parallel computing nature. **(3)** We propose self-coverage, cross-coverage, and fusion-coverage to refine the attention weights using the past alignment information from the current and previous layers. **(4)** Experiments demonstrate the effectiveness of our proposed CoMER model. Specifically, we achieve new state-of-the-art performance on the CROHME 2014/2016/2019 test sets using a single CoMER model, reaching 59.33%/59.81%/62.97% in ExpRate.

We believe that our proposed attention refinement framework not only works for handwritten mathematical expression recognition. Our ARM can help refine the attention weights and improve the alignment quality for all tasks that require dynamic alignment. To this end, we intend to extend ARM in the transformer as a general framework for solving various vision and language tasks in the future work (e.g., machine translation, text summarization, image captioning).

Acknowledgements. This work is supported by the projects of National Key R&D Program of China (2019YFB1406303) and National Nature Science Foundation of China (No. 61876003), which is also a research achievement of Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

1. Alvaro, F., Sánchez, J.A., Benedí, J.M.: Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. *Pattern Recognition Letters* **35**, 58–67 (2014)
2. Anderson, R.H.: Syntax-directed recognition of hand-printed two-dimensional mathematics. In: *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*. pp. 436–459 (1967)
3. Bengio, Y., Frasconi, P., Simard, P.: The problem of learning long-term dependencies in recurrent networks. In: *IEEE international conference on neural networks*. pp. 1183–1188. IEEE (1993)
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: *European Conference on Computer Vision* (2020)
5. Chan, K.F., Yeung, D.Y.: Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition* **3**(1), 3–15 (2000)
6. Cheng, Z., Bai, F., Xu, Y., Zheng, G., Pu, S., Zhou, S.: Focusing attention: Towards accurate text recognition in natural images. In: *Proceedings of the IEEE international conference on computer vision*. pp. 5076–5084 (2017)
7. Cornia, M., Stefanini, M., Baraldi, L., Cucchiara, R.: Meshed-memory transformer for image captioning. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 10578–10587 (2020)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: *North American Chapter of the Association for Computational Linguistics* (2018)
9. Ding, H., Chen, K., Huo, Q.: An encoder-decoder approach to handwritten mathematical expression recognition with multi-head attention and stacked decoder. In: *International Conference on Document Analysis and Recognition*. pp. 602–616. Springer (2021)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021*. OpenReview.net (2021), <https://openreview.net/forum?id=YicbFdNTTy>
11. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: *International Conference on Artificial Intelligence and Statistics* (2011)
12. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: *Computer Vision and Pattern Recognition* (2017)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv: Learning* (2015)
14. Lavirotte, S., Pottier, L.: Mathematical formula recognition using graph grammar. In: *Document Recognition V. vol. 3305*, pp. 44–52. International Society for Optics and Photonics (1998)
15. Li, Z., Jin, L., Lai, S., Zhu, Y.: Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. pp. 175–180. IEEE (2020)

16. Liu, L., Utiyama, M., Finch, A., Sumita, E.: Agreement on target-bidirectional neural machine translation. In: North American Chapter of the Association for Computational Linguistics (2016)
17. Luo, Y., Ji, J., Sun, X., Cao, L., Wu, Y., Huang, F., Lin, C., Ji, R.: Dual-level collaborative transformer for image captioning. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. pp. 2286–2293. AAAI Press (2021), <https://ojs.aaai.org/index.php/AAAI/article/view/16328>
18. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Empirical Methods in Natural Language Processing (2015)
19. MacLean, S., Labahn, G.: A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. *International Journal on Document Analysis and Recognition (IJDAR)* **16**(2), 139–163 (2013)
20. Mahdavi, M., Zanibbi, R., Mouchere, H., Viard-Gaudin, C., Garain, U.: 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). pp. 1533–1538. IEEE (2019)
21. Mouchere, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). In: 2014 14th International Conference on Frontiers in Handwriting Recognition. pp. 791–796. IEEE (2014)
22. Mouchère, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions. In: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 607–612. IEEE (2016)
23. Pan, Y., Yao, T., Li, Y., Mei, T.: X-linear attention networks for image captioning. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020. pp. 10968–10977. Computer Vision Foundation / IEEE (2020). <https://doi.org/10.1109/CVPR42600.2020.01098>, https://openaccess.thecvf.com/content_CVPR_2020/html/Pan_X-Linear_Attention_Networks_for_Image_Captioning_CVPR_2020_paper.html
24. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Neural Information Processing Systems (2019)
25. Rosendahl, J., Herold, C., Petrick, F., Ney, H.: Recurrent attention for the transformer. In: Proceedings of the Second Workshop on Insights from Negative Results in NLP. pp. 62–66 (2021)
26. 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: 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 181–186. IEEE (2020)
27. Tu, Z., Lu, Z., Liu, Y., Liu, X., Li, H.: Modeling coverage for neural machine translation. In: Meeting of the Association for Computational Linguistics (2016)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Neural Information Processing Systems (2017)

29. 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* pp. 1–16 (2020)
30. 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: *International conference on machine learning*. pp. 2048–2057 (2015)
31. Zanibbi, R., Mouchère, H., Viard-Gaudin, C.: Evaluating structural pattern recognition for handwritten math via primitive label graphs. In: *Document Recognition and Retrieval* (2013)
32. Zhang, J., Du, J., Dai, L.: Multi-scale attention with dense encoder for handwritten mathematical expression recognition. In: *2018 24th international conference on pattern recognition (ICPR)*. pp. 2245–2250. IEEE (2018)
33. Zhang, J., Du, J., Yang, Y., Song, Y.Z., Wei, S., Dai, L.: A tree-structured decoder for image-to-markup generation. In: *ICML*. p. In Press (2020)
34. 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)
35. Zhao, W., Gao, L., Yan, Z., Peng, S., Du, L., Zhang, Z.: Handwritten mathematical expression recognition with bidirectionally trained transformer. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) *16th International Conference on Document Analysis and Recognition, ICDAR 2021, Lausanne, Switzerland, September 5-10, 2021, Proceedings, Part II. Lecture Notes in Computer Science*, vol. 12822, pp. 570–584. Springer (2021). https://doi.org/10.1007/978-3-030-86331-9_37