Local and Global Graph Modeling with Edge-weighted Graph Attention Network for Handwritten Mathematical Expression Recognition

Yejing XIE[a], Richard Zanibbi[b], Harold Mouchère[a]

[a]*Nantes Universite, Ecole Centrale Nantes, CNRS, LS2N, UMR 6004, Nantes, F-44300, France*
[b]*Document and Pattern Recognition Lab, Rochester Institute of Technology, Rochester, New York, USA*

Abstract

In this paper, we present a novel approach to Handwritten Mathematical Expression Recognition (HMER) by leveraging graph-based modeling techniques. We introduce an end-to-end model with an Edge-weighted Graph Attention Mechanism (EGAT), designed to perform simultaneous node and edge classification. This model effectively integrates node and edge features, facilitating the prediction of symbol classes and their relationships within mathematical expressions. Additionally, we propose a stroke-level Graph Modeling method for both local (LGM) and global (GGM) information, which applies the end-to-end model to Online HMER tasks, transforming the recognition problem into node and edge labeling tasks in graph structure. By capturing both local and global graph features, our method ensures comprehensive understanding of the expression structure. Through the combination of these components, our system demonstrates superior performance in symbol detection, relation classification, and expression-level recognition.

Keywords: Online Handwritten Mathematical Expression Recognition, Graph Attention Network, Graph-to-Graph Model, Graph Modeling

1. Introduction

Mathematical expressions (ME) [2] are essential in scientific research, engineering, education, and many other fields. Unlike structured but less intuitive editing tools and markup languages (e.g., LaTeX), handwritten ME (HME) [3] are more user-friendly for humans but harder for machines to recognize due to variations in writing styles and habits. Handwritten Mathematical Expression Recognition (HMER) converts handwritten math into markup language for easier processing and rendering, offering broad application potential but facing significant challenges. Compared with Optical Character Recognition (OCR) [4], HMER must handle not only handwriting variability but also the complex 2D structure of mathematical notation. HMER can be classified into Online and Offline modes: Offline data are static images from scanners, cameras, or smartphones, while Online data are sequences of temporal trajectories captured by digital devices (e.g., tablets, pens), segmented into strokes based on pen-down and pen-up events. In this work, we focus on Online HMER because it retains temporal and stroke-level details, where both the shape of each stroke and their spatial relationships provide valuable information for recognition. Compared to pixel-based Offline data, Online data contain less redundant information and enable faster processing.

Existing deep learning architectures for HMER are typically based on encoder-decoder models as illustrated in the upper part of Fig. 1, the comprehensive review of which is provided in Section 2.1. This structure failed to leverage the graph structure inherent in mathematical layouts, making it difficult to capture and use the relationships between symbols. Moreover, they operate in a latent space that is not directly aligned with the input data, such as individual strokes. Motivated by these limitations, we further explore a graph-based representation of HME for end-to-end stroke-level recognition of Online data, leveraging large-scale stroke-level annotated datasets [5], illustrated in the lower part of Fig. 1.

Graph structure plays an important role in HMER [6], as both input HMEs and their output ME representations can be effectively modeled as graph structures, and Graph Neural Networks (GNNs) [7] are well-suited for processing graph-structured data. In particular, ME representations can be represented as Stroke Label Graphs (SLGs) [8] with stroke-level annotations, where nodes correspond to strokes belonging to specific symbols, and edges capture the spatial relations between these strokes. Compared to sequential markup languages such as LaTeX, SLGs provide an explicit, interpretable representation that directly preserves the 2D layout.

In our previous work [1], we proposed a stroke-level graph labeling approach that jointly embeds node and edge features using an Edge-weighted Graph Attention Mechanism. Unlike conventional methods that focus solely on node features, this approach enables unified prediction of both node and edge attributes in a single pass. Experiments demonstrated that the model effectively fuses features and captures structural information, although its overall performance at the expression level remains limited. To address this, in this work we improve the end-to-end model with novel message passing and feature fusion strategies, combined with advanced optimization techniques, and extend graph modeling from local to global levels. The key contributions of this paper are as follows:

- We enhance the end-to-end model with an **Edge-weighted Graph Attention Mechanism (EGAT)**, explicitly designed for joint node and edge classification by integrating fused message passing, message concatenation, residual connections, and auxiliary readout mechanisms.
- Building on our earlier **Local Graph Modeling (LGM)**, which focuses on fine-grained local relationships between strokes, we extend to **Global Graph Modeling (GGM)**, applying a general EGAT to Online HMER as a graph labeling task, thereby leveraging both local and global graph features.
- We propose a detailed evaluation of our system which achieves strong performance in symbol detection, relation classification, and expression level recognition.

The rest of the paper is organized as follows. Section 2 reviews related work on both HMER and Graph Neural Networks (GNNs). Section 3 presents the proposed methodology, including the EGAT-based end-to-end model (Section 3.1) and stroke-level graph modeling for HMER (Section 3.2). Experimental results are reported in Section 4, followed by conclusions in Section 5.

2. Related Work

In this section, we review related work on Math Recognition and Graph Neural Networks.

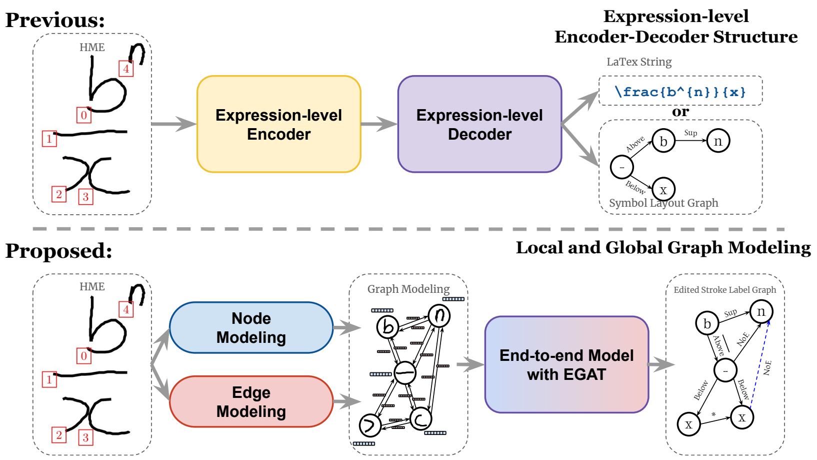


Figure 1: The overview of the proposed end-to-end model. In constrast to encoder-decoder structures, our approach directly aligns the strokes of Online HME with their corresponding final graph representation of the expression.

2.1. Math Recognition

In the early years, HMER algorithms were mainly Structural Recognition Methods, which construct a graph or tree with explicit relations between symbols, and then match the graph or tree using predefined rules. Some grammar-based methods [9, 10] and graph-based methods [11, 12] recognize the HME by their expression structure. These methods allow integration of prior knowledge about mathematical expressions into the recognition process, such as spatial relationships and grammar constraints.

In recent years, HMER has increasingly relied on deep learning, enabled by high-performance computing, which minimizes the need for handcrafted features and grammar rules while substantially improving recognition accuracy. A common paradigm is the encoder-decoder architecture, where the encoder extracts visual features from the input expression and the decoder generates its symbolic representation. Most works build upon two seminal encoder architectures: (1) Track, Attend and Parse (TAP) [13], which employs a GRU-based encoder to capture sequential dependencies between strokes in Online HME, (2) Watch, Attend and Parse (WAP) [14], which uses a CNN-based encoder to extract deep visual features from Offline HME images. Almost all of these approaches rely on encoders to extract features in latent space, without explicitly considering the structural characteristics of mathematical expressions. Exceptions include [15] which constructs a graph from Online strokes and [16] which constructs a graph from YOLOv5-detected symbols and extracted visual features from Offline images. Both utilize a Graph Attention Network (GAT) to encode the resulting representation. Moreover, [17] proposed a multi-stage pre-training strategy, including localization-based visual pre-training and LaTeX-based decoder pre-training via cross-modal graph alignment. Such approaches do not combine nodes and edges features during message passing. so that they fail to combine spatial positional relationships between the strokes and strokes' shape information, which are crucial for accurately capturing the general structure of MEs.

Many studies have focused on decoder design in HMER. Early works such as WAP and TAP primarily employed GRU-based decoders to model sequential dependencies. With the advent of the Transformer [18], subsequent approaches have increasingly adopted Transformer-based decoders, which offer improved modeling of long-range dependencies and parallelizable computation [19]. Several common strategies in deep learning have been applied to enhance the decoder of HMER models, including adversarial learning [20], attention refinement [21], contrastive learning for semantic invariance [22]. To better capture the structural characteristics of ME, some researches integrated tree structures into the decoder, such as handcrafted path extraction rules [23], syntactic information incorporation [24], sequential relation decoder [25, 26], tree structure prediction scoring [27], branch-parallel decoder [28], and position forest auxiliary recognition [29]. These improvements introduce structural information into the decoder, but they still operate in a sequential manner, which is not optimal for capturing the inherently 2D structure of mathematical expressions. With the recent popularity of large language models (LLMs) and vision LLMs (vLLMs), several approaches have emerged that fine-tune vLLMs for HMER tasks [30, 31]. While effective, they heavily rely on massive datasets and extremely large numbers of model parameters, and largely overlook the interpretability of mathematical expressions, making it difficult to align encoder features with decoder semantics, such as relying too much on in-context learning, rather than explicitly recognizing individual symbols and their relationships.

2.2. Graph Neural Networks

In recent work in HMER, graphs are increasingly used to represent the structural relationships between handwritten strokes and symbols. Graph Neural Networks (GNNs) [32] have emerged as a powerful tool for handling such graph-structured data. A key mechanism in GNNs is message passing [33], where nodes in the graph exchange information with their neighbors. Each node aggregates messages from its connected nodes to update its own features, allowing the model to capture both local information. However, standard GNNs propagate messages using only node features, limiting the integration of edge information. Edges influence the process solely through their presence or absence, without accounting for associated weights or features.

An extension is the Graph Attention Network (GAT) [34], which improves the message passing process by assigning different attention weights to each edge, allowing the model to focus more on important relationships between nodes. While GAT effectively emphasizes key edges for feature aggregation, its attention mechanism is still limited when edges carry high-dimensional or complex topological information, making it challenging to fully exploit rich edge representations.

Additionally, the concept of a master node [35] can be introduced in graph models to enhance the capture of global information. The master node is connected to all other nodes, aggregating the information from the entire graph and enabling a more comprehensive understanding of the graph structure. Although similar in spirit to the CLS token [36] in Transformer-based models, which also serves as a global aggregator, the master node is an explicit graph element whose connectivity and message passing mechanism are impacted by the graph's topology, rather than pairwise interactions in a sequence.

In order to address the limitations of encoder-decoder architectures and node-only GNNs, the proposed end-to-end model integrates message passing, attention mechanisms, and master nodes, enabling the joint exploitation of node and edge information to capture both structural and semantic patterns in graph-based data. The model design is presented in the following sections.

3. Methodology

As shown in Fig. 1, our approach consists of two main components. Section 3.1 introduces a general end-to-end graph learning framework that takes graphs with both node and edge features as input. Through message passing and feature fusion, it predicts all node and edge labels in a single forward pass. Section 3.2 presents a modeling strategy tailored for HMER, where HMEs are represented as graphs with node and edge features, and the ground truth is reformulated into a graph with the same structure. This design ensures a direct correspondence between predicted and target elements.

3.1. End-to-end Model with Edge-weighted Graph Attention Mechanism

GNNs are well-suited for modeling graph structures and relational dependencies, making them ideal for tasks like HMER where both node-embedded stroke information and edge-embedded relations are crucial. To fully exploit edge information, we propose an end-to-end model with node and edge embeddings, edge-weighted graph attention for message passing and feature integration, and readout modules for simultaneous node and edge classification.

The model performs a graph-to-graph transformation task, which convert a model graph $\mathbf{G}$, containing both node and edge features, into a target ground truth graph $\hat{\mathbf{G}}$ with node and edge labels.

As shown in Eq.(1), a general graph is defined by 4 parts.

$$\mathbf{G} = (\mathbf{V}, \mathcal{A}, \mathbf{H}, \mathbf{B}) \tag{1}$$

where $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ is the set of nodes in the graph, with n representing the number of nodes. $\mathcal{A} \in \{0, 1\}^{n \times n}$ is the adjacency matrix, where each element $\mathcal{A}_{ij}$ represents the presence or absence of an edge between nodes $\mathbf{v}_i$ and $\mathbf{v}_j$. The embedded node features $\mathbf{H} \in \mathbb{R}^{n \times d_1}$, where $\mathbf{h}_i \in \mathbb{R}^{1 * d_1}$ is a column vector representing node $\mathbf{v}_i$, and d_1 is the dimension of node features. The embedded edge features are $\mathbf{B} \in \mathbb{R}^{n \times n \times d_2}$, where $\mathbf{b}_{ij} \in \mathbb{R}^{1 * d_2}$ is a column vector representing edge between node $\mathbf{v}_i$ and $\mathbf{v}_j$, and d_2 is the dimension of edge features. $\mathbf{b}_{ij}$ is "null" if there is no edge between node $\mathbf{v}_i$ and $\mathbf{v}_j$, meaning $\mathcal{A}_{ij} = 0$.

In Section 3.2, the construction of graph $\mathbf{G}$ will be detailed in the context of HMER, where the node features represent shape information of strokes,edge features represent spatial relationships between strokes, and graph connectivity is determined using a Line-of-Sight (LOS) graph[12].

$\hat{\mathbf{G}}$ is the ground truth graph, which match all the nodes and edges labels with the corresponding nodes and edges features in the graph $\mathbf{G}$ is defined in Eq.(2).

$$\hat{\mathbf{G}} = \left(\mathbf{V}, \mathcal{A}, \hat{\mathbf{H}}, \hat{\mathbf{B}}\right) \tag{2}$$

Nodes set $\mathbf{V}$ and adjacency matrix $\mathcal{A}$ are of the same type in both $\mathbf{G}$ and $\hat{\mathbf{G}}$. This assumes that the underlying graph structure remains unchanged in the transformation process. $\hat{\mathbf{H}} \in \{0, 1, \ldots, C_1 - 1\}^n$ is the nodes labels, where $\hat{\mathbf{h}}_i$ is the label of node $\mathbf{v}_i$, and C_1 is the number of classes for node classification tasks. $\hat{\mathbf{B}} \in \{0, 1, \ldots, C_2 - 1\}^{n \times n}$ is the edge labels, where $\hat{\mathbf{b}}_{ij}$ is the label of edge between node $\mathbf{v}_i$ and $\mathbf{v}_j$, and C_2 is the number of classes for edge classification tasks.

In Section 3.2.3, the construction of ground truth graph $\hat{\mathbf{G}}$ for HMER will be detailed. where the graph can be generated by edited Stroke Label Graph (ESLG) with stroke-level annotations of HME.

3.1.1. Baseline Structure

As shown in Fig. 2, node features and edge features are first embedded separately using the Node Embedding Module $\mathcal{E}_{node}$ and Edge Embedding Module $\mathcal{E}_{edge}$. According to the specific characteristics of node and edge features, different neural networks may be used for node and edge embedding modules. The node and edge embedding features $\mathbf{H}^0$ and $\mathbf{B}^0$ are the input to the Edge-weighted Graph Attention Module $\mathcal{G}$ for message passing and feature integration. This is a key idea in end-to-end models, and it will be detailed in Section 3.1.2. After Q layers of EGAT for node and edge feature fusion, the fused node features $\mathbf{H}^Q$ and edge features $\mathbf{B}^Q$ are utilized for node and edge classification tasks, respectively, with the assistance of the Node Readout $\mathcal{R}_{node}$ and Edge Readout $\mathcal{R}_{edge}$. Both Readout networks for node features and edge features are Multi-Layer Perceptron (MLP) with similar structure. The last layer of the Readout networks is a softmax layer for the classification tasks. The output of the whole end-

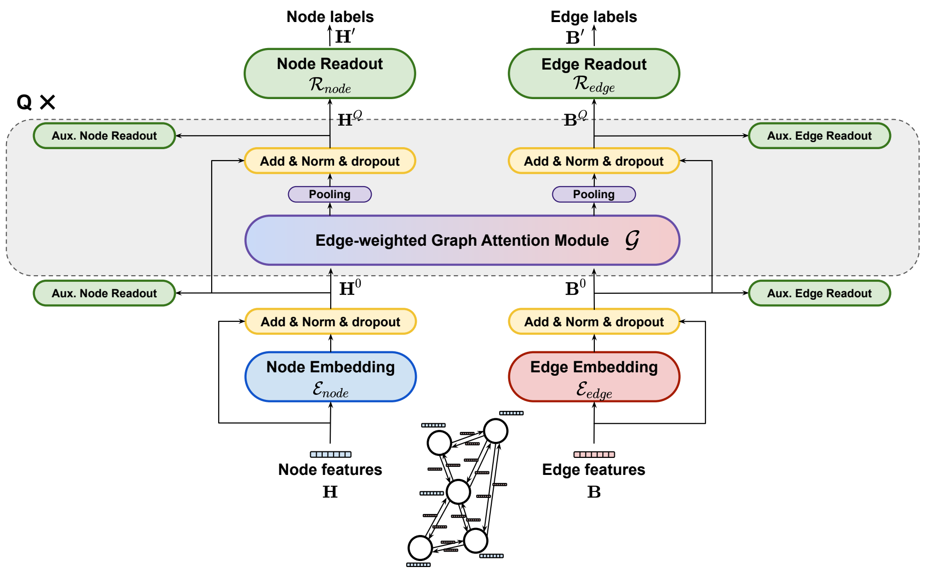
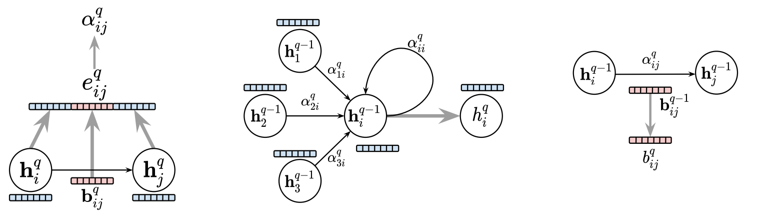


Figure 2: The baseline structure of the proposed end-to-end model.

to-end model is the probability distribution of node labels $\mathbf{H}' \in \mathbb{R}^{n\times C_1}$ and edge labels $\mathbf{B}' \in \mathbb{R}^{n\times n\times C_2}$. To enhance the model's performance, we incorporate advanced strategies like residual connections and auxiliary readout, building on the baseline structure. These will be explained in detail in Section 3.1.3.

3.1.2. Edge-weighted Graph Attention Module

Standard node-centric message passing often fails to capture the intricate dependencies in graphs with complex edge attributes. To address this, we extend the vanilla GAT into an Edge-weighted Graph Attention Mechanism (EGAT), which integrates both node and edge features into the Attention Weight Computation (as shown in Fig.3a) and Message Passing (Fig.3b and 3c) stages. This bidirectional enhancement allows node information to guide edge updates, while edge features concurrently refine node representations. Furthermore, we introduce Message Concatenation (shown in Fig.4) as a further fusion step to explicitly combine these updated signals, ensuring a more comprehensive integration of the graph's structural and featural information within an end-to-end framework.



(a) Attention weight. (b) Node Message Passing. (c) Edge Message Passing.

Figure 3: Attention Weight Computation and Message Passing, from layer $q-1$ to layer q using attention α_{ij}^q.

Attention Weight Computation. For layer q, when we calculate the attention weights, in addition to the concatenation of corresponding neighbor node features, we also concatenate the edge features together by Eq. (3).

$$e_{ij}^q = a^\top \left[W_h^q \cdot \mathbf{h}_i^q \oplus W_b^q \cdot \mathbf{b}_{ij}^q \oplus W_h^q \cdot \mathbf{h}_j^q \right] \tag{3}$$

where W_h and W_b are learnable weights for node features and edge features, while a is a learnable attention coefficient, and $\oplus$ indicates the concatenation operation.

Then, using the softmax function, we get the attention weights α_{ij} from Eq.(4).

$$\alpha_{ij}^q = \text{softmax}(e_{ij}^q) = \frac{exp(e_{ij}^q)}{\sum_{k \in \mathcal{N}_i} exp(e_{ik}^q)} \tag{4}$$

which indicates the importance of the edge between node i and node j in the q-th layer by softmax normalization.

Message Passing. In the q-th layer, the node features h_i^q are updated by weighted sums of neighbor features, as shown in Eq.(5),

$$h_i^q = \sum_{j \in \mathcal{N}_i} \left(\alpha_{ij}^q \cdot W_h^q \cdot \mathbf{h}_j^{q-1} \right) \tag{5}$$

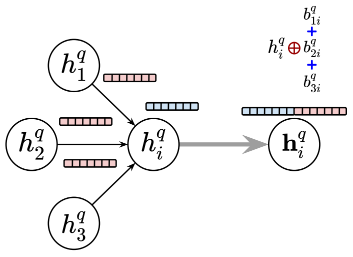
$\mathbf{h}_j^{q-1}$ is all the neighbor node features in last layer, while $j \in \mathcal{N}_i$ means all the neighbor nodes of node i, and also the corresponding attention weights α_{ij}^q.

Likewise, the edge features b_{ij}^q are be updated using Eq.(6),

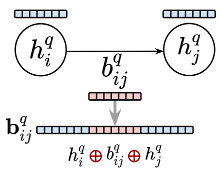
$$b_{ij}^q = \alpha_{ij}^q \cdot W_b^q \cdot \mathbf{b}_{ij}^{q-1} \tag{6}$$

where α_{ij} isthe corresponding attention weights and b_{ij}^{q-1} is the edge features in last layer.

Message Concatenate. In our design, nodes and edges act as mutual indicators during message passing. Relying solely on attention scores is insufficient to capture the complex relationships in mathematical expressions. Specifically, edge features should be informed by the two nodes they connect, while node features should reflect the specific way they are linked to their neighbors. Therefore, we update both nodes and edges using information from both sources. To further integrate these features, a message concatenation strategy is applied, allowing the model to combine local details with the overall graph structure more effectively.



(a) Node Message Concatenation



(b) Edge Message Concatenation

Figure 4: Message Concatenation.

The q-th node features $\mathbf{h}_i^q$ will be calculated by the concatenation of node features h_i^q with the sum of all the connected edge features b_{ij}^q, as shown in Eq.(7) and Fig. 4a,

$$\mathbf{h}_i^q = h_i^q \oplus \sum_{j \in \mathcal{N}_i} b_{ij}^q \tag{7}$$

where $j \in \mathcal{N}_i$, and $\oplus$ indicates the concatenation operation.

The q-th edge features $\mathbf{b}_{ij}^q$ will be calculated by Eq.(8) and Fig. 4b.

$$\mathbf{b}_{ij}^q = h_i^q \oplus b_{ij}^q \oplus h_j^q \tag{8}$$

which is the concatenation of edge features b_{ij}^q with the connected 2 node features h_i^q and h_j^q.

There is a mismatch between the number of parameters in the front and back layers caused by the concatenation, which will lead to an explosion in the number of parameters. To avoid this, a pooling layer is added after the message concatenation operation to match the dimension of the features.

3.1.3. Advanced Optimization Techniques

Residual Connections. Residual connections, as known as Shortcuts [37], have been proven to be effective in the training of deep neural networks, which can help to avoid the vanishing gradient problem and also enhance the deep model performance. We applied the residual connections after Embedding module and also after each EGAT module. Subsequent experiments have shown that the residual connections can effectively improve the performance of baseline model both in node and edge classification tasks. Besides, dropout layers are also added after the residual connections, which can help to avoid overfitting and improve the generalization of the model.

Auxiliary Readout. Inspired by GoogLeNet [38], we add auxiliary classifiers to intermediate layers to improve convergence. Specifically, Readout Modules are placed after the Embedding Module and each Edge-weighted Graph Attention Module layer, serving as auxiliary classifiers for node and edge labeling. These Auxiliary Readouts, identical in structure to the final Readout (MLP with softmax), provide intermediate supervision that enhances prediction accuracy. Their inclusion transforms the original bi-objective optimization into a multi-objective one, with the corresponding loss described in Section 3.1.4.

3.1.4. Multi-Objective Optimization

Node and Edge Classification. The node and edge classification tasks are both multi-class classification tasks. Cross-entropy loss is widely used in multi-class classification tasks. However, for different tasks, more advanced loss functions can be applied to improve the performance of the model.

In math recognition, the edge classification is not balanced, so we applied the cross-entropy loss $\mathcal{L}_n$ for node classification, and also the focal loss $\mathcal{L}_e$ for edge classification task, which can help to focus on the hard samples and also the unbalanced samples.

The final optimization loss $\mathcal{L}$ is a weighted sum of node classification loss $\mathcal{L}_n$ and edge classification loss $\mathcal{L}_e$, as shown in Eq.(9).

$$\mathcal{L} = \lambda_1 \mathcal{L}_n + (1 - \lambda_1)\mathcal{L}_e \tag{9}$$

where λ_1 is the weight factor.

Auxiliary Loss. Auxiliary loss is used to supervise the intermediate Auxiliary Readout, which can help the model to converge faster and more stable for the entire model. We calculate the auxiliary loss $\mathcal{L}^i_{aux}$ for the i^{th} Auxiliary Readout, which is the same as the node classification loss and edge classification loss, as shown in Eq.(10).

$$\mathcal{L}^i_{aux} = \lambda_1 \mathcal{L}^i_n + (1 - \lambda_1)\mathcal{L}^i_e \tag{10}$$

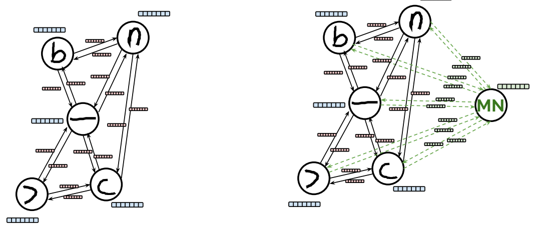
The multi-objective optimization loss $\mathcal{L}$ is a weighted sum of node classification loss, edge classification loss and also the auxiliary loss, as shown in Eq.(11).

$$\mathcal{L} = \lambda_1 \mathcal{L}_n + (1 - \lambda_1)\mathcal{L}_e + \sum_{i=1}^{M} \lambda_2 \mathcal{L}^i_{aux} \tag{11}$$

where the weight factor λ_2 is set for the auxiliary loss, and as well as the weight factor λ_1 is set for the node classification loss and edge classification loss, which is the same for each layer for simplicity. M is the number of Auxiliary Readout in the whole end-to-end structure.

3.2. Graph Modeling for HMER

We introduce the Graph Modeling method for HMER, which transfers the Online HME into a graph structure, and then applies the proposed graph-to-graph model for the node and edge classification tasks. The Graph Modeling method is based on the Stroke-level Graph Modeling, each node represents a stroke in Online HME, and each edge represents the position relation between two strokes. The fundamental Stroke-level Graph Modeling is detailed in section 3.2.1, which captures the local information of the Online HME. However, it misses the global information of the whole expression. To address this, we propose a Global Graph Modeling method in section 3.2.2, which is built on the Local Graph Modeling, to capture the full structure of the entrie expression. Local and Global Graph Modeling for HMER is shown in Fig. 5.



(a) Local Graph Modeling. (b) Global Graph Modeling with master node (MN).

Figure 5: Illustration of the Local and Global Graph Modeling for HMER.

3.2.1. Local Graph Modeling

Our Stroke-level Graph Modeling which effectively captures the local information of Online HME, is denoted to Local Graph Modeling (LGM) $\mathbf{G}^L$. This is a similar modeling method as our previous work [1]. Each stroke $\mathbf{v}_i^L \in \mathbf{V}^L$ is represented by a node in Online HME. It also requires detailed embedding representation of node features $\mathbf{H}^L$, edge features $\mathbf{B}^L$, and the connectivity $\mathcal{A}^L$ between the nodes, without considering the global information combination of the full expression.

Graph Connectivity Construction. While a fully-connected graph offers maximum flexibility, it introduces excessive redundant noise and irrelevant dependencies that hinder effective learning. To achieve a sparse but sufficiently detailed topology, we employ the Line of Sight (LOS)[39] technique to establish connections based on the visibility between stroke convex hulls. This geometric prior effectively prunes redundant noise compared to fully-connected graphs. To further enrich the representation, we incorporate temporal adjacencies to form a LOS+t graph, ensuring both spatial visibility and stroke writing orders are captured. Experimental results confirm that this constrained representation consistently outperforms the fully-connected baseline, providing a more robust prior for the subsequent attention mechanism.

Thus, the adjacency matrix $\mathcal{A}^L \in \{0, 1\}^{n\times n}$ provides the connectivity relationships between nodes, where n is the number of strokes in Online HME. All the edges in graph $\mathbf{G}^L$ are bidirectional edges, that means $\mathcal{A}_{ij}^L = \mathcal{A}_{ji}^L$.

Node Modeling. Only the x and y coordinates of each stroke are used as node features. However, due to variations in data collection devices and user writing habits, stroke coordinates for the same symbol class can differ significantly. To mitigate these variations, standardization is necessary. We applied a strategy to remove writing speed effects [40] from the raw stroke coordinates, along with Gaussian normalization according to average diagonal length of all strokes of the expression. The standardized stroke features $\mathbf{h}_i \in \mathbb{R}^{2\times d_n}$ of node $\mathbf{v}_i^L \in \mathbf{V}^L$, where d_n is the number of sampling points per stroke for node modeling. The matrix $\mathbf{H}^L \in \mathbb{R}^{n\times 2\times d_n}$ can represent all the individual stroke features of the whole Online HME.

Edge Modeling. Edge modeling requires a strategy for modeling or extracting the relations features between strokes. Previous Online document analysis work extracted multi-dimensional geometric features from stroke bounding boxes, such as [41]. Rather than considering the geometric relationships only between stroke bounding boxes, we aim to capture more detailed shape and bi-directional position information. In this study, we apply a Fuzzy Relative Positioning Template (FRPT) [42] for relations extraction.

The main idea is to calculate the radian degree between standard vectors $\overrightarrow{e^x}$ in 4 directions (Right→, Left←, Up↑ and Down↓) and the vector $\overrightarrow{OP_k}$ from the center of initial stroke O to the sampling points of target stroke P_k, as shown in Eq.(12).

$$\theta_k^x = max\left(0, 1 - \frac{2}{\pi}\arccos\left(\frac{\overrightarrow{OP_k}\cdot\overrightarrow{e^x}}{\sqrt{|\overrightarrow{OP_k}|^2 + |\overrightarrow{e^x}|^2}}\right)\right) \tag{12}$$

The connections between the nodes always have 2 directions with different embeddings. Besides the spatial relations, we also consider the distance D_k between the two sampling points of the initial and target strokes, as shown in Eq.(13).

$$\mathbf{b}_{ij} = \left[\theta_0^{\rightarrow},\ldots,\theta_{d_2}^{\rightarrow},\theta_0^{\leftarrow},\ldots,\theta_{d_2}^{\leftarrow},\theta_0^{\uparrow},\ldots,\theta_{d_2}^{\uparrow},\theta_0^{\downarrow},\ldots,\theta_{d_2}^{\downarrow},D_0,\ldots,D_{d_2}\right] \tag{13}$$

where the edge features $\mathbf{b}_{ij} \in \mathbb{R}^{5d_2}$ of edge $\mathbf{e}_{ij}^L \in \mathbf{E}^L$ are represented by the concatenation of the 4 directions and the distance, where d_2 is the number of sampling points per stroke for edge modeling. $b_{ij} \neq b_{ji}$ for the bi-directional edge, since the initial and target strokes are different. Thus, the matrix $\mathbf{B}^L \in \mathbb{R}^{n \times n \times 5d_2}$ represents the relationship features between strokes.

Sub-Expression Splitting and Masking. In real-world applications, the number of strokes in Online HME varies, with complex expressions containing many strokes and simple ones containing only a few. We split the Online HME into multiple sub-expressions $\mathbf{G}^{sub}$ if the stroke count exceeds N_{max}. Expressions or sub-expressions with fewer than N_{max} strokes are padded with a special blank stroke, where all coordinates are set to 0. Node features $\mathbf{H}^{sub} \in \mathbb{R}^{N_{max} \times d_1}$ and edge features $\mathbf{B}^{sub} \in \mathbb{R}^{N_{max} \times N_{max} \times d_2}$ are constructed for each sub-expression.

The sub-expression splitting is done solely based on the order of stroke input, without considering grammar rules, spatial relations, or symbolic integrity. As a result, multi-stroke symbols may be split across different sub-expressions, leading to incomplete symbols that complicate recognition. To address this, we mask incomplete symbols within sub-expressions during training. If $\exists \mathbf{v}_i \in \mathbf{V}^{sub}$, $\exists \mathbf{v}_j \notin \mathbf{V}^{sub}$ and the relation label $\hat{\mathbf{b}}_{ij}$ between $\mathbf{v}_i$ and $\mathbf{v}_j$ is "*", that means the symbol is split across sub-expressions. We mask both the incomplete node features $\mathbf{h}_i^{sub}$, the relative edge features $\mathbf{b}_{ik}^{sub}$ and also their ground-truth labels $\hat{\mathbf{h}}_i^{sub}$ and $\hat{\mathbf{b}}_{ik}^{sub}$, where $\mathcal{A}_{ik} = 1$. These nodes and edges features are not involved in the training loss calculation and backward propagation, ensuring only complete symbols are used for training. The splitting and masking strategy is applied exclusively during training to enable multi-batch processing, while the validation and testing stages use the original full graph modeling. This approach enhances the model's local features representations, improving the recognition of individual symbols and their relations. However, focusing too much on local information risks overlooking the global structure of the whole expression.

3.2.2. Global Graph Modeling

To capture the global information of the full expression, we proposed a simple global graph modeling method by adding a master node to the local graph modeling.

Master Node and Edge Connection. In terms of interpretability, deep GNNs can capture information from neighbors at increasing depths. However, relying solely on deep networks makes it challenging to effectively aggregate high-level global information.

To address this, the concept of a master node has proven effective in graph-based models for capturing global information, particularly in deep networks. Therefore, we introduced a master node $\mathbf{v}_M$ in global graph modeling, which is connected to all nodes in the original local graph modeling, as shown in Fig. 5b. The nodes set of the global graph $\mathbf{V}^G = \mathbf{V}^L \cup \mathbf{v}_M$, master node $\mathbf{v}_M$ is connected to all the local nodes $\mathbf{v}_i^L \in \mathbf{V}^L$, so the adjacency matrix $\mathcal{A}^G \in \{0,1\}^{(n+1)\times(n+1)}$ is constructed as Eq.(14),

$$\mathcal{A}_{ij}^G = \begin{cases} 1 & \text{if } i = 0 \text{ or } j = 0 \text{ or } \mathcal{A}_{(i-1)(j-1)}^L = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

where $i = 0$ or $j = 0$ means the master node $\mathbf{v}_M$ is involved in the connection, and the other edge connection remain unchanged from the local graph modeling.

Through message passing, the virtual master node automatically aggregates global information from all local nodes, and the information retrieved at each layer is propagated to subsequent layers.

Feature Initialization. The initial features of the master node and the connected edges should be taken into account. To better integrate the shape information from all the strokes in the entire expression, rather than sub-expression split by 3.2.1, the initial features of the master node $\mathbf{h}_M$ are the summation of all the node features in the original modeled graph, which denotes $\mathbf{h}_M = \sum_{\mathbf{v}_i \in \mathbf{V}^L} \mathbf{h}_i$ and $\mathbf{H}^G = [\mathbf{h}_M, \mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_n]^T$.

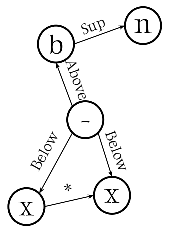
The edge features in the global graph $\mathbf{B}^G \in \mathbb{R}^{(n+1)\times(n+1)\times 5d_2}$ are initialized as Eq.(15).

$$\mathbf{B}_{ij}^G = \begin{cases} \mathbf{B}_{(i-1)(j-1)}^L & \text{if } i \neq 0 \text{ and } j \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

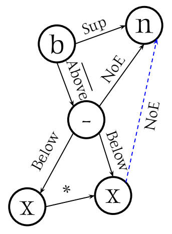
where the connected edges between the master node and the local nodes are initialized as vector 0 to avoid spatial confusion, and the other edges remain unchanged from the local graph modeling.

3.2.3. *Edited Stroke Label Graph*

The results of HMER can be represented as different formats, such as MathML tree, LaTeX string, and also the stroke-level graph called Stroke Label Graph (SLG)



(a) SLG



(b) ESLG

Figure 6: From Stroke Label Graph (SLG) to Edited-SLG (ESLG) for expression $\frac{b^n}{x}$ written with 5 strokes. "Below", "Above", "Sup" are the positional relations, "$\overline{\text{Above}}$" is the opposite positional relation of "Above". While "*" means belonging relation and "NoE" means no relation.

proposed by [8], which is a directed graph structure, where the nodes represent the strokes belonging to which symbol and the edges represent the relations between the strokes, Fig 6a shows an example of Stroke Label Graph. Based on SLG, we make some simple edge modifications to match the nodes and edges of SLG with the modelised stroke-level graph, which is denoted as Edited Stroke Label Graph (ESLG), $\hat{\mathbf{G}}$ in Eq.(2). The relations in mathematical expression are all directional according to the relative position. For example, "b" is above "–" in the expression $\frac{b^n}{x}$, so the edge "above" should point from "–" to "b". In LOS+t graph, all the edges are bidirectional, the high dimension features of edge "–" to "b" should represent an "Above" relationship, while the edge "b" to "–" should represent the opposite of "Above", noted as "$\overline{\text{Above}}$". However, labeling both directions of bidirectional edges can lead to confusion in the final decision process. To resolve this, we retain only one directed edge for each pair of connected strokes, following the writing order. This means each edge is directed from the earlier stroke to the later stroke, as calculated in Eq.(16).

$$\hat{\mathcal{A}}_{ij} = \begin{cases} 1 & \text{if } j > i \text{ and } \mathcal{A}_{ij} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

This strategy maintains the structure of the SLG and disregards syntactic rules, but avoids confusion of directions through temporal ordering. The edge between "–" and "b" is kept as "$\overline{\text{Above}}$" from "b" to "–" according to the writing order. In addition, the "*" relation, which indicates that two strokes belong to the same symbol, and the

"NoE" (No Edge) relation, which labels edges present in the LOS+t graph but absent in the SLG. An example of the edited SLG is illustrated in Fig. 6b. Thus, ESLG $\hat{\mathbf{G}}$ has the same node set $\mathbf{V}$ as the corresponding modeling graph $\mathbf{G}$, while $\hat{\mathbf{H}} \in \mathbb{R}^{n \times C_1}$ is the node labels of entire Online HME belonging to $C_1 = C_n$ classes of symbols, and $\hat{\mathbf{B}} \in \mathbb{R}^{n \times n \times C_2}$ is the edge labels of entire Online HME belonging to $C_2 = 2C_e + 2$ classes of relations including C_e positional relations, C_e opposite positional relations, "*" and "NoE".

4. Experiments

In this section, we present the experimental results of the proposed end-to-end framework with Edge-weighted Graph Attention Network for HMER. This includes LGM-EGAT for Local Graph Modeling and GGM-EGAT for Global Graph Modeling.

4.1. Experiments Setup

4.1.1. Datasets

The proposed model is trained and evaluated on the newest Competition on Recognition of Online Handwritten Mathematical Expression dataset (CROHME 2023) [5]. The CROHME 2023 provides large-scale Online HME data in InkML [1] format, and provides the ground truth with stroke-level annotations in LaTeX, MathML and SLG format, including 101 symbol classes ($C_n = 101$) and 6 relation classes ("Right", "Sup", "Sub", "Above", "Below", "Inside"), where $C_e = 6$ are labeled in the SLG.

4.1.2. Implementation Details

Our model is implemented in PyTorch-Lightning and trained on a single NVIDIA GeForce RTX 2080 Ti GPU. Hyperparameters are optimized via Optuna, with initial learning rate $\eta = 0.00027$, batch size $B = 32$, maximum strokes per sub-expression $N_{\max} = 16$, bi-objective loss weight $\lambda_1 = 0.5$, auxiliary loss weight $\lambda_2 = 0.3$, focal loss parameter $\gamma = 1.5$, and dropout $p = 0.1$. Training uses the Adam optimizer with a learning rate decay of 0.1 after 20 stagnant epochs and stops at 200 epochs. Node

[1] http://www.w3.org/2003/InkML

and edge embeddings use $d_1 = 150$ and $d_2 = 10$ sampling points, respectively, with configurations in Table 1. XceptionTime [43] is used for node embedding, and a simple MLP for edge embedding.

Table 1: End-to-end Model Configuration.

Module Name	**Details**	**Module nb.**
Node Embedding	$\mathcal{E}_n$ = XceptionTime(2, 150)	1
Edge Embedding	$\mathcal{E}_e$ = MLP(50, 384, 512)	1
EGAT Layer	$\mathcal{G}_{node} = (512, 512), \mathcal{G}_{edge} = (512, 512)$	5
Node Readout	$\mathcal{R}_n$ = MLP(1024, 384, 101)	1
Edge Readout	$\mathcal{R}_e$ = MLP(1024, 384, 14)	1
Aux. Node Readout	$\mathcal{R}_n^{aux}$ = MLP(1024, 384, 101)	5
Aux. Edge Readout	$\mathcal{R}_e^{aux}$ = MLP(1024, 384, 14)	5

4.2. Results

In this section, we present enriched ablation studies for the proposed end-to-end model, and compare the performance of the proposed model with the state-of-the-art methods.

4.2.1. Ablation Study

We conducted a series of experiments to evaluate the performance of the proposed HMER system with different end-to-end model structures, graph modeling strategies and graph connectivity construction methods. All of the experiments are evaluated on the validation set of CROHME 2023 to avoid overfitting and information leakage. The metrics are the accuracy of classification (Clf. Acc.) for both nodes and edges at the stroke (primitive) level. The correct symbol segmentation rate (Seg.), the correct symbol segmentation and symbol classification rate (Sym.), the correct relation classification rate (Rel.), the correct expression recognition rate (Exp.) and the correct structure recognition rate (Struc.) are expression-level accuracies provided by LgEval[44].

As shown in Table 2, in order to compare the effectiveness of the proposed end-to-end model, we applied the global graph modeling strategy and "LOS+t" as the graph connectivity construction method. The baseline model is the end-to-end model only with node and edge embedding network, EGAT layers, and node and edge readout networks, which was proposed in our previous work [1]. "GNN" replaces EGAT layers with standard GNN layers using basic message passing, while "GAT" replaces them with standard GAT layers, computing node and edge features separately without feature fusion in attention score calculation or message passing. Based on this baseline model, we added the residual connection (BL+Shortcut), auxiliary loss (BL+Aux. Loss), and also the message concatenate (BL+Mes. Cat.) we proposed in the previous sections. The proposed model is the baseline model with all of advanced strategies. The baseline EGAT model substantially outperforms the standard GNN and GAT models, benefiting from edge-weighted attention fusion.

Moreover, the message concatenation strategy substantially boosts edge classification, while the auxiliary loss and residual connections improve both node and edge classification. Overall, the proposed model achieves the highest accuracy in both tasks, confirming the effectiveness of the advanced strategies.

Table 2: Ablation Study for end-to-end model and graph modelisation. 95% confidence interval for Exp. via bootstrap resampling is reported for all settings. The **Proposed** model is significantly improved compared to baseline model ($p < 0.05$ based on McNemar's test with Exp. metric).

	Clf. Acc.		**Exp. level**				
End2end	**Node**	**Edge**	**Seg.**	**Sym.**	**Rel.**	**Exp.**	**Stru.**
GNN	68.69	82.36	72.12	62.21	70.45	12.56 ± 1.91	45.89
GAT	70.88	84.61	72.98	63.12	74.45	14.12 ± 2.01	46.83
Baseline (EGAT)	92.42	95.60	97.54	90.88	90.60	45.30 ± 2.82	71.50
BL + Shortcut	93.21	95.89	97.56	91.63	91.29	48.18 ± 2.83	72.91
BL + Aux. Loss	93.05	95.79	97.77	91.66	91.25	47.83 ± 2.82	72.33
BL + Mes. Cat.	91.30	96.80	98.13	89.60	92.20	42.95 ± 2.82	75.56
Proposed	**94.40**	**97.50**	**98.39**	**93.34**	**93.69**	**55.88 ± 2.79**	**80.73**

In Table 3, we applied both local and global graph modeling strategies, and compared the performance of our new model with the baseline model. According to node and edge classification accuracy, the global graph modeling strategy slightly improves both node and edge classification performance. Although the segmentation accuracy, symbol recognition accuracy, and relation recognition accuracy have not improved substantially even less, the expression recognition and structure recognition accuracy in expression level. This suggeststhat the global graph modeling strategy can enhance the global information capture and improve the recognition of the entire expression, which is beneficial for the expression recognition and structure recognition tasks.

Table 3: Ablation Study for Graph Modeling Strategies. "BL" is Baseline end2end model proposed in [1], "Proposed" is the proposed model in this paper with message concatenate and advanced strategies. "G.M" is the graph modeling strategy, while "L." is local graph modeling, "G." is global graph modeling. 95% confidence interval for Exp. via bootstrap resampling is reported for all settings. The Proposed model with global graph modeling (**G.**) is significantly improved compared to local graph modeling (**L.**) ($p < 0.05$ based on McNemar's test with Exp. metric).

		Clf. Acc.		**Exp. level**				
End2end	**G.M**	**Node**	**Edge**	**Seg.**	**Sym.**	**Rel.**	**Exp.**	**Stru.**
BL	L.	92.12	94.78	97.56	90.20	90.55	44.89 ± 2.80	71.62
BL	G.	92.42	95.60	97.54	90.88	90.60	45.30 ± 2.83	71.50
Proposed	L.	94.16	97.48	**98.44**	**93.45**	**93.79**	55.23 ± 2.79	79.85
Proposed	**G.**	**94.40**	**97.50**	98.39	93.34	93.69	**55.88 ± 2.79**	**80.73**

In Table 4, we explored the effectiveness of the graph connectivity connection. Compared with the Full Connected (FC) graph, the proposed "LOS+t" strategy improves performance in all the evaluation metrics, over the weaker fusion model in the baseline model. How the proposed "LOS+t" strategy can help the end-to-end model learn more specific and accurate graph structure information, which is beneficial for the expression and structure recognition tasks. Meanwhile, the proposed end2end model with strong information fusion ability is able to learn the complex graph connectivity connection relation, and could achieve better performance with the help of "LOS+t" strategy.

4.2.2. Comparison with State-of-the-Art

In this section, we compare our proposed LGM-EGAT (Local Graph Modeling with Edge-weight Graph Attention Mechanism) and GGM-EGAT (Global Graph Modeling with Edge-weight Graph Attention Mechanism) systems with state-of-the-art methods. Since there are only a few methods focusing on Online handwritten mathematical expression recognition, we also take some Offline methods into consideration. For fair comparison, we fine-tuned and evaluated our proposed models on the CROHME 2016

Table 4: Ablation Study for Graph Connectivity Connection. "BL" is Baseline end2end model [1], "Proposed" is the new model with message concatenation and advanced strategies. "G.C.C." is the graph connectivity construction, while "FC" is Full Connected, "LOS+t" is the proposed strategy. 95% confidence interval for Exp. via bootstrap resampling is reported for all settings. **LOS+t** graph connectivity construction is significantly improved compared to Full Connected graph ($p < 0.05$ based on McNemar's test with Exp. metric).

		Clf. Acc.		**Exp. level**				
End2end	**G.C.C.**	**Node**	**Edge**	**Seg.**	**Sym.**	**Rel.**	**Exp.**	**Stru.**
BL	FC	82.11	91.16	96.30	79.07	86.27	22.50 ± 2.43	58.34
BL	LOS+t	92.42	95.60	97.54	90.88	90.60	45.30 ± 2.83	71.50
Proposed	FC	93.88	97.12	98.05	92.84	92.31	52.00 ± 2.93	76.62
Proposed	LOS+t	**94.40**	**97.50**	**98.39**	**93.34**	**93.69**	**55.88 ± 2.79**	**80.73**

and CROHME 2019 data sets. The results are presented in Table 5. The proposed LGM-EGAT and GGM-EGAT systems outperform the state-of-the-art methods on both CROHME 2016 and CROHME 2019 datasets, including traditional encoder-decoder methods [14, 13], tree-based encoder-decoder methods[45, 23, 17], and also encoded-decoder methods with graph neural networks [15, 16]. LGM-EGAT and GGM-EGAT reduce the error rate by 10.00% and 10.67%, respectively, compared to the G2G model [15].

All listed models except VLPG†[17] are trained and evaluated on the same dataset without any additional data augmentation or pre-training strategies.

In Table 6, we compare our proposed LGM-EGAT and GGM-EGAT systems with the top-ranked methods in the CROHME 2023 competition. According to the competition reports [46, 5], most of the top-ranked methods are trained with additional data or use ensemble strategies to improve performance. Our models are trained only on the official training data in CROHME2023 without any additional data. Due to this difference in training data, our models do not outperform the top-ranked methods in expression recognition accuracy. But the proposed models still achieve comparable performance, especially in structure recognition accuracy.

Table 5: Comparison with State-of-the-Art. "On." is model with Online data, "Off." is model with Offline data. "-" means the result is not available from the original paper. The metric for both CROHME 2016 and CROHME 2019 is the expression recognition accuracy. "†" means the model is pretrained with additional data. 95% confidence interval for Exp. via bootstrap resampling is reported for Baseline [1], LGM-EGAT and GGM-EGAT. GGM-EGAT is significantly improved compared to baseline model ($p < 0.05$ based on McNemar's test with Exp. metric).

Systems	**Data Type**	**CROHME 2016**	**CROHME 2019**
WAP[14]	Off.	44.45	-
GETD[16]	Off.	55.27	54.13
VLPG[17]	Off.	54.40	56.99
VLPG†[17]	Off.	60.51†	62.34†
TAP[13]	On.	44.80	-
Tree-construction[23]	On.	41.76	-
Tree-BiLSTM[45]	On.	27.03	-
G2G[15]	On.	52.05	-
Baseline[1]	On.	43.87 ± 2.86	48.29 ± 2.95
LGM-EGAT	On.	**56.41** ± **2.88**	**58.22** ± **2.83**
GGM-EGAT	On.	**56.67**± **2.88**	**60.72** ± **2.75**

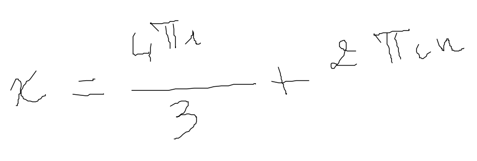
Table 6: Comparison with CROHME Competition 2019 and 2023. All the results are for CROHME 2019 and 2023. "Exp." at expression level is the expression recognition rate, while "≤ 1" and "≤ 2 " are the expression recognition rate with less than 1 and 2 errors, respectively. The "Stru." is the structure recognition accuracy."*" denotes an ensemble of several differently initialized recognition models. "†" denotes the model is trained with additional data. "‡" denotes using a mathematical Language Model and additional LaTeX sequences.

2019	Exp.	≤ 1	≤ 2	Stru.	**2023**	Exp.	≤ 1	≤ 2	Stru.
iFLYTEK*‡	80.73	88.99	90.74	91.49	Sunia†	82.34	90.26	92.47	92.41
Samsung†	79.82	87.82	89.15	89.32	YP_OCR*	72.55	83.57	86.22	86.60
MyScript†‡	79.15	86.82	89.82	90.66	TUAT	41.10	54.52	60.04	56.85
PAL-v2*‡	62.55	74.98	78.40	79.15	DPRL	38.19	53.39	58.39	59.98
MathType†	60.13	74.40	78.57	79.15	-	-	-	-	-
TUAT	39.95	52.21	56.54	58.22	-	-	-	-	-
LGM-EGAT	58.22 ± 2.83	70.89	75.48	83.07	LGM-EGAT	53.04 ± 2.03	66.26	72.22	78.39
GGM-EGAT	60.72 ± 2.75	71.14	76.73	83.74	GGM-EGAT	55.30 ± 2.00	68.43	72.91	79.13

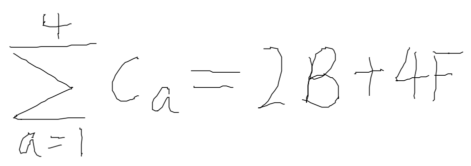
4.3. Analysis

4.3.1. Attention Visualization

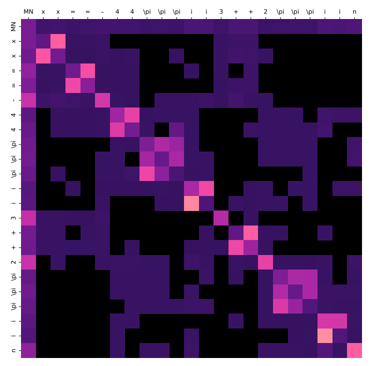
We save the attention scores of the last EGAT layer, which usually represent the importance of the neighbors of one node, and also the importance of the connected edges. We visualize the attention scores α_{ij} from the final EGAT layer as an $N \times N$



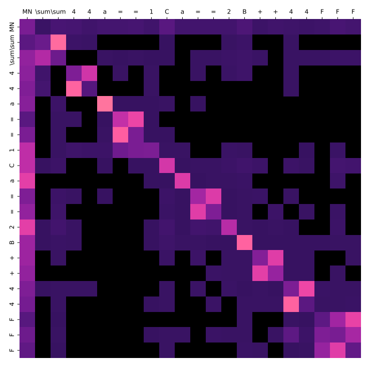
(a) Ex.1 $x = \frac{4\pi_i}{3} + 3\pi_i n$



(b) Ex.2 $\sum_{a=1}^{4} C_a = 2B + 4F$



(c) Attention visualization of Ex.1



(d) Attention visualization of Ex.2

Figure 7: Attention Visualization Examples. The lighter color represents the higher attention score, which means the more important the neighbor node or the connected edge. The darker color is the opposite, and the pure black means no connected edge between the nodes.

heatmap (Fig. 7c and 7d) to interpret the learned relationships between strokes. Higher attention intensities are predominantly clustered among strokes belonging to the same symbol, such as "π", "i", "x", and "+" in Fig. 7c, as well as "$\sum$" and "F" in Fig. 7d. These patterns demonstrate that the model effectively captures local structural dependencies, which is crucial for accurate expression segmentation.

Furthermore, the Master Node exhibits a distinct interaction pattern. The first column of the heatmap consistently shows high intensity, indicating that the Master Node aggregates global context from all constituent strokes. Conversely, the first row reveals that the Master Node's global information selectively influences specific nodes, such as "*C*" and "*B*" in Fig. 7d. We conjecture that the Master Node provides essential global cues to resolve shape ambiguities, such as distinguishing between uppercase and lowercase "*C*", thereby significantly enhancing classification robustness.

4.3.2. Effect of Expression Length

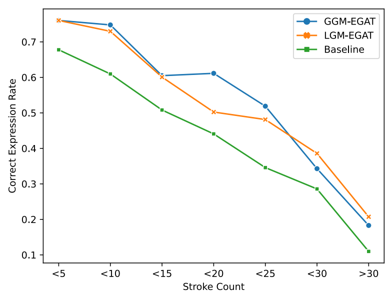
Expression Length can express the complexity of the math expression. We plotted charts to analyze the expression recognition rate for different expression lengths by strokes count and symbol count, as shown in Fig. 8. In general, the model performs better on shorter expressions', the recognition rate decreases as the expression length increases. The proposed GGM-EGAT model and LGM-EGAT model improve over the baseline model, both in simple and complex expressions. In addition, the GGM-EGAT model and LGM-EGAT model have a similar performance in simple expressions. The GGM-EGAT model with global information has better performance with the increase of expression complexity, especially on the 2023 dataset, which highlights the benefit of global information for the recognition of complex expressions.

4.3.3. Error Analysis

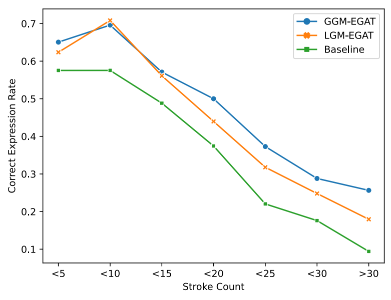
Table 7 summarizes the confusion histograms for the CROHME 2023 Test set, highlighting the top four most challenging symbols and symbol pairs for the GGM-EGAT model. Regarding individual symbols, misclassifications primarily stem from morphological similarities, such as among "*x*", "*X*", and "$\times$". While the Master Node and neighboring context mitigate these ambiguities, subtle errors persist. For symbol pairs, errors frequently arise from incorrect edge predictions. For instance, the "||" pair often fails to achieve a correct "Right" relation. This suggests that without explicit structural constraints, edge-level predictions remain susceptible to relational ambiguities even when individual symbols are correctly identified.

5. Conclusion

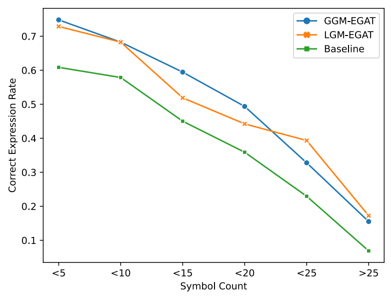
In this paper, we presented a novel stroke-level graph modeling approach for Online HMER. By transforming HMER into a joint node and edge classification task,



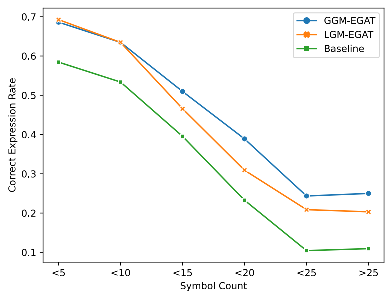
(a) 2019 Testset



(b) 2023 Testset



(c) 2019 Testset



(d) 2023 Testset

Figure 8: Effect of Expression Length. 8a and 8b are the correct expression rate of different expression lengths by strokes count. 8c and 8d are the file correct expression rate of different expression lengths by symbol count.

our method effectively captures both local structural dependencies and global context. We introduced an end-to-end architecture that facilitates the deep fusion of node and edge features, ensuring a cohesive representation of mathematical expressions. Experimental results on the CROHME 2019 and 2023 datasets demonstrate substantial performance gains, notably achieving a 60.72% and 55.30% expression-level recognition rate, respectively, and outperforming the previous baseline [1] by a substantial margin.

Future work will focus on extending this framework through a "Stroke-to-Sequence" strategy to handle more complex datasets and exploring "Graph-to-Stroke" generation for synthetic data augmentation. Furthermore, we intend to investigate the adaptability of our graph-based approach in other 2D structural domains, such as chemical notation

Table 7: Error Analysis of Symbols and Symbol Pairs with the proposed GGM-EGAT model in CROHME 2023 Testset. **E** is the symbol or symbol pair for recognition, # is the number of occurrences in the 2023 test set. $\mathbf{err_1}$, $\mathbf{err_2}$, $\mathbf{err_3}$, $\mathbf{err_4}$ are the most confusing mistakes of this symbol. Only the occurrences greater than 3 are listed. "$\Vert$" in Symbol Pairs means missing the relation between 2 symbols.

		E	**#**	$\mathbf{err_1}$	**#**	$\mathbf{err_2}$	**#**	$\mathbf{err_3}$	**#**	$\mathbf{err_4}$	**#**
Symbols	1st	1	133	$,$	15	$($	13	11	12	$\prime$	10
	2nd	x	67	X	24	$\times$	8	k	5	u	5
	3rd	n	58	m	20	u	8	4	4	x	4
	4th	$-$	47	1	4	$--$	4	$\sqrt{}$	3	$+$	3
Symbol Pairs	1st	$x)$	24	$X)$	10	$x \Vert)$	3	-	-	-	-
	2nd	$\overline{2}$	22	-2	3	-	-	-	-	-	-
	3rd	$(x$	18	$(X$	9	-	-	-	-	-	-
	4th	$\underline{1}$	20	$1 \Vert -$	5	$\underline{2}$	3	-	-	-	-

recognition and musical score analysis, to further validate its versatility.

References

[1] Y. Xie, H. Mouchère, Stroke-level graph labeling with edge-weighted graph attention network for handwritten mathematical expression recognition, in: E. H. Barney Smith, M. Liwicki, L. Peng (Eds.), Document Analysis and Recognition - ICDAR 2024, Springer Nature Switzerland, Cham, 2024, pp. 38–55.

[2] R. Zanibbi, D. Blostein, Recognition and retrieval of mathematical expressions, International Journal on Document Analysis and Recognition (IJDAR) 15 (4) (2012) 331–357.

[3] U. Garain, B. B. Chaudhuri, Recognition of online handwritten mathematical expressions, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (6) (2004) 2366–2376.

[4] R. Mithe, S. Indalkar, N. Divekar, Optical character recognition, International journal of recent technology and engineering (IJRTE) 2 (1) (2013) 72–75.

[5] Y. Xie, H. Mouchère, F. Simistira Liwicki, S. Rakesh, R. Saini, M. Nakagawa, C. T. Nguyen, T.-N. Truong, Icdar 2023 crohme: Competition on recognition

of handwritten mathematical expressions, in: International Conference on Document Analysis and Recognition, Springer, 2023, pp. 553–565.

[6] T.-N. Truong, C. T. Nguyen, R. Zanibbi, H. Mouchère, M. Nakagawa, A survey on handwritten mathematical expression recognition: The rise of encoder-decoder and gnn models, Pattern Recognition 153 (2024) 110531.

[7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems 32 (1) (2020) 4–24.

[8] H. Mouchère, C. Viard-Gaudin, D. H. Kim, J. H. Kim, U. Garain, Icfhr 2012 competition on recognition of on-line mathematical expressions (crohme 2012), in: 2012 International Conference on Frontiers in Handwriting Recognition, IEEE, 2012, pp. 811–816.

[9] R. Yamamoto, S. Sako, T. Nishimoto, S. Sagayama, On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar, in: Tenth international workshop on frontiers in handwriting recognition, Suvisoft, 2006.

[10] A.-M. Awal, H. Mouchère, C. Viard-Gaudin, A global learning approach for an online handwritten mathematical expression recognition system, Pattern Recognition Letters 35 (2014) 68–77, frontiers in Handwriting Processing.

[11] H.-J. Winkler, M. Lang, Online symbol segmentation and recognition in handwritten mathematical expressions, in: 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 4, IEEE, 1997, pp. 3377–3380.

[12] L. Hu, R. Zanibbi, Mst-based visual parsing of online handwritten mathematical expressions, in: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2016, pp. 337–342.

[13] J. Zhang, J. Du, L. Dai, Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition, IEEE Transactions on Multimedia 21 (1) (2018) 221–233.

[14] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, L. Dai, Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition, Pattern Recognition 71 (2017) 196–206.

[15] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, C.-L. Liu, Graph-to-graph: towards accurate and interpretable online handwritten mathematical expression recognition, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 2925–2933.

[16] J.-M. Tang, H.-Y. Guo, J.-W. Wu, F. Yin, L.-L. Huang, Offline handwritten mathematical expression recognition with graph encoder and transformer decoder, Pattern Recognition 148 (2024) 110155.

[17] H.-Y. Guo, C. Wang, F. Yin, X.-H. Li, C.-L. Liu, Vision–language pre-training for graph-based handwritten mathematical expression recognition, Pattern Recognition 162 (2025) 111346.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[19] W. Zhao, L. Gao, Z. Yan, S. Peng, L. Du, Z. Zhang, Handwritten mathematical expression recognition with bidirectionally trained transformer, in: Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16, Springer, 2021, pp. 570–584.

[20] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, C.-L. Liu, Handwritten mathematical expression recognition via paired adversarial learning, International Journal of Computer Vision 128 (2020) 2386–2401.

[21] W. Zhao, L. Gao, Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition, in: European conference on computer vision, Springer, 2022, pp. 392–408.

[22] Q. Lin, X. Huang, N. Bi, C. Y. Suen, J. Tan, Cclsl: Combination of contrastive learning and supervised learning for handwritten mathematical expression recognition, in: Proceedings of the Asian Conference on Computer Vision, 2022, pp. 3724–3739.

[23] T.-N. Truong, H. T. Nguyen, C. T. Nguyen, M. Nakagawa, Learning symbol relation tree for online handwritten mathematical expression recognition, in: C. Wallraven, Q. Liu, H. Nagahara (Eds.), Pattern Recognition, Springer International Publishing, Cham, 2022, pp. 307–321.

[24] Y. Yuan, X. Liu, W. Dikubab, H. Liu, Z. Ji, Z. Wu, X. Bai, Syntax-aware network for handwritten mathematical expression recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4553–4562.

[25] J. Zhang, J. Du, Y. Yang, Y.-Z. Song, S. Wei, L. Dai, A tree-structured decoder for image-to-markup generation, in: International Conference on Machine Learning, PMLR, 2020, pp. 11076–11085.

[26] S. Peng, L. Gao, K. Yuan, Z. Tang, Image to latex with graph neural network for mathematical formula recognition, in: Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16, Springer, 2021, pp. 648–663.

[27] J. Zhu, W. Zhao, Y. Li, X. Hu, L. Gao, Tamer: Tree-aware transformer for handwritten mathematical expression recognition, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 39, 2025, pp. 10950–10958.

[28] Z. Li, W. Yang, H. Qi, L. Jin, Y. Huang, K. Ding, A tree-based model with branch parallel decoding for handwritten mathematical expression recognition, Pattern Recognition 149 (2024) 110220.

[29] T. Guan, C. Lin, W. Shen, X. Yang, Posformer: recognizing complex handwritten mathematical expression with position forest transformer, in: European Conference on Computer Vision, Springer, 2024, pp. 130–147.

[30] H.-Y. Guo, F. Yin, J. Xu, C.-L. Liu, Hie-vl: A large vision-language model with hierarchical adapter for handwritten mathematical expression recognition, in: ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2025, pp. 1–5.

[31] Y. Li, J. Jiang, J. Zhu, S. Peng, B. Wei, Y. Zhou, L. Gao, Uni-muMER: Unified multi-task fine-tuning of vision-language model for handwritten mathematical expression recognition, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[32] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE transactions on neural networks 20 (1) (2008) 61–80.

[33] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, in: International conference on machine learning, PMLR, 2017, pp. 1263–1272.

[34] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, et al., Graph attention networks, stat 1050 (20) (2017) 10–48550.

[35] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, in: International Conference on Learning Representations, 2018.

[36] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.

[37] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[39] L. Hu, R. Zanibbi, Line-of-sight stroke graphs and parzen shape context features for handwritten math formula representation and symbol segmentation, in: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2016, pp. 180–186.

[40] M. Pastor, A. Toselli, E. Vidal, Writing speed normalization for on-line handwritten text recognition, in: Eighth International Conference on Document Analysis and Recognition (ICDAR'05), IEEE, 2005, pp. 1131–1135.

[41] J.-Y. Ye, Y.-M. Zhang, Q. Yang, C.-L. Liu, Contextual stroke classification in online handwritten documents with graph attention networks, in: 2019 International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2019, pp. 993–998.

[42] A. Delaye, E. Anquetil, Fuzzy relative positioning templates for symbol recognition, in: 2011 International Conference on Document Analysis and Recognition, IEEE, 2011, pp. 1220–1224.

[43] E. Rahimian, S. Zabihi, S. F. Atashzar, A. Asif, A. Mohammadi, Xceptiontime: independent time-window xceptiontime architecture for hand gesture classification, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 1304–1308.

[44] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, 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, IEEE, 2014, pp. 791–796.

[45] T. Zhang, H. Mouchere, C. Viard-Gaudin, Tree-based blstm for mathematical expression recognition, in: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Vol. 1, IEEE, 2017, pp. 914–919.

[46] M. Mahdavi, R. Zanibbi, H. Mouchère, C. Viard-Gaudin, U. 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), IEEE, 2019, pp. 1533–1538.