

Link prediction Graph Neural Networks for structure recognition of Handwritten Mathematical Expressions

Cuong Tuan Nguyen¹[0000-0003-2556-9191], Ngoc Tuan Nguyen¹[0009-0002-8306-6269], Triet Hoang Minh Dao¹[0009-0001-1404-5926], Huy Minh Nhat Nguyen¹[0009-0007-5539-1124], and Huy Truong Dinh¹[0000-0003-0290-6685]

Vietnamese-German University, Ho Chi Minh City, Vietnam
{cuong.nt2,huy.td}@vgu.edu.vn,
{21024003,10423179,10423045}@student.vgu.edu.vn

Abstract. We propose a Graph Neural Network (GNN)-based approach for Handwritten Mathematical Expression (HME) recognition by modeling HMEs as graphs, where nodes represent symbols and edges capture spatial dependencies. A deep BLSTM network is used for symbol segmentation, recognition, and spatial relation classification, forming an initial primitive graph. A 2D-CFG parser then generates all possible spatial relations, while the GNN-based link prediction model refines the structure by removing unnecessary connections, ultimately forming the Symbol Label Graph. Experimental results demonstrate the effectiveness of our approach, showing promising performance in HME structure recognition.

Keywords: GNN · EGAT · math · handwriting recognition

1 Introduction

Mathematical notations are essential in scientific documents for conveying concepts in mathematics and physics. With the emergence of digital pens and tablets, there is an increasing trend toward using handwritten mathematical notations as input, leading to a demand for systems that recognize these handwritten mathematical expressions (HMEs). We refer to the problem of online handwriting recognition, where each input sample is a sequence of trajectories captured by each pen-down called handwritten strokes.

Recognizing HMEs presents a challenging problem due to the complex two-dimensional structure of mathematical expressions. Early works focus on using structural parsing methods with a context-free grammar to represent mathematical expressions [10,17,4], tree structure networks [13], or graph-based algorithms [2]. However, due to the separated tasks of symbol segmentation, symbol recognition and structural analysis, these methods have limited overall expression recognition rate.

With the rise of deep learning algorithms, encoder-decoder approaches solve the problem of separated tasks by an end-to-end learning model. The encoder-decoder models developed from Recurrent Neural Networks (RNNs) decoder with attention [12,11], to the transformer decoder [15,14] and multi-task learning [16]. These techniques focus on recognizing HME images and outputting one-dimensional character sequences in LaTeX format. However, they often fail to explicitly represent the hierarchical and spatial characteristics inherent to mathematical notations. Without clear symbol segmentation and two-dimensional relationships, these methods have limited effectiveness in practical scenarios that require structural clarity for computational manipulation and interactive editing.

Recent advancements have leveraged graph neural networks (GNNs) to address these limitations. Wu et al. [8] proposed a graph-to-graph (G2G) learning method using stroke-level representations, modeling handwriting strokes as nodes and spatial relationships as edges. Xie et al. [9] expanded this concept with an Edge Graph Attention Network (EGAT) focused on stroke-level interactions. Although stroke-level methods capture detailed structural information, they face the complexity of the graph due to numerous stroke primitives involved.

We propose a symbol-level graph recognition method for handwritten math expression recognition to overcome these limitations. By shifting from stroke-level to symbol-level representation, our approach reduces structural complexity of the graph. Our method integrates several key components:

First, we employ a Bidirectional Long Short-Term Memory (BLSTM) network for simultaneous symbol segmentation, recognition, and initial spatial relationship classification. Processing online handwritten data, this approach uses global contextual information to improve recognition of three tasks.

Next, we utilize a two-dimensional Cocke-Younger-Kasami (CYK) parsing algorithm to explore all possible spatial relationships among recognized symbols, constructing a symbol-level graph with recognized symbols as nodes and recognized relations as edges.

Finally, we further refine structural relationships using an Edge-featured Graph Attention Network (EGAT) on symbol-level graphs for a link prediction task. EGAT link prediction predicts whether an edge should be removed to produce the Symbol Layout Tree (SLT).

In summary, the primary contributions of this paper are:

- Formulate the problem of mathematical structure recognition to the problem of building symbol-level graph and link prediction.
- Propose a deep BLSTM and 2D CYK parser to construct symbol-level graph.
- Apply EGAT model for symbol-level link prediction to output mathematical expression as an SLT.

The remainder of this paper is structured as follows: Section 2 reviews related works, Section 3 describes our proposed methodology in detail, Section 4 presents experimental results and analysis, and Section 5 concludes with key insights and future research directions.

The source code for the experiments will be available in https://github.com/ntcuong2103/math_online_egat

2 Related works

Handwritten Mathematical Expression (HME) recognition has evolved over recent decades, progressing from grammar-driven and heuristic-based methods to sophisticated deep learning and Graph Neural Network (GNN)-based approaches. The central challenge remains accurately parsing the complex, hierarchical two-dimensional structure inherent in mathematical expressions.

2.1 Graph-Based Structural methods

Hu and Zanibbi [2] formulate the problem of mathematical expression recognition as a Minimum Spanning Tree (MST) parsing by employing Edmonds’ algorithm on directed Line-Of-Sight (LOS) graphs [1]. Their MST-based parser efficiently constructs expression trees by selecting edges that optimize spatial relationships among symbols, considerably reducing computational complexity and improving parsing accuracy compared to traditional CYK-based methods. Although MST approaches delivered promising improvements, their success heavily depended on accurate symbol segmentation and preliminary edge prediction.

Truong et al. used rule based split and merge methods for constructing the SLT [7]. The method benefits from the global context of an HME for symbol and relation classification [5].

2.2 Graph Neural Network-Based methods

Recent advancements have shifted towards deep learning-based models, particularly those leveraging Graph Neural Networks (GNNs), due to their intrinsic capability of modeling structured data. GNNs directly encode structural relationships within a graph representation, exploiting local context through iterative message-passing mechanisms to refine node and edge predictions.

Stroke-Level GNN Stroke-level methods treat individual pen strokes as graph primitives, bypassing explicit symbol segmentation. Wu et al. [8] introduced a pioneering Graph-to-Graph (G2G) model, conceptualizing HME recognition as graph transformation from raw stroke primitives to recognized symbol and relation graphs. Their approach employed sub-graph attention mechanisms to align input strokes precisely with their corresponding recognized symbols, simultaneously addressing segmentation and structural recognition tasks.

Building upon stroke-level concepts, Xie et al. [9] proposed the Edge-weighted Graph Attention Network (EGAT) for HME structure recognition. EGAT integrates edge features into attention computations, directly predicting symbol identities and spatial relations from stroke graphs. Although stroke-level GNN methods are advantageous in reducing error propagation from segmentation stages, they inherently produce highly dense graphs, significantly increasing computational demands.

Symbol-Level GNN Symbol-level GNN models address stroke-level complexity by pre-segmenting strokes into symbols. Tang et al. [6] utilized a Graph Reasoning Network (GRN), structuring detected symbols from HME image into an SLT. GRN methods significantly enhance recognition accuracy by reducing the graph size and emphasizing meaningful symbol interactions, thus providing an efficient trade-off between segmentation complexity and graph representation fidelity.

Summary Our method builds a symbol-level graph of recognized symbols and spatial relations between symbols, then applies GNN to predict whether an edge should be removed from the graph. The approach is similar to the MST, but we use GNN instead of Edmond’s algorithm to remove edges. We also propose using the 2D CYK algorithm to build the symbol-level graph instead of the LOS algorithm.

3 Methodology

In this section, we propose a handwritten mathematical expression recognition method that constructs a symbol-level graph as input to the GNN for link prediction to refine the connections.

3.1 Overview of the method

The pipeline of our method is shown in Fig. 1. A sequence of extracted features from an online HME input is fed to a deep BLSTM to predict symbols and relations. The segmented symbols are input to a symbol-level parser to build the symbol-level graph. Finally, the symbol-level graph is input to GNN for link prediction and produces SLT output.

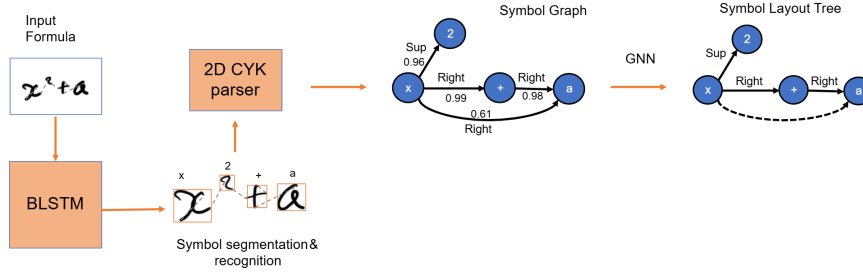


Fig. 1. Overview of the method

3.2 Build symbol level graph

This consists of two main stages as shown in Fig. 2: (1) symbol segmentation, recognition, and relation detection using a temporal recognition BLSTM network, and (2) symbol-level graph construction via a 2D CYK parsing algorithm. The BLSTM outputs, at each time step corresponding to a recognized symbol or pair of symbols, a probability distribution over the relation classes. These probabilities seed our primitive graph construction.

Symbol and Relation Recognition This stage utilizes a deep bidirectional long short-term memory (BLSTM) network to process handwritten mathematical expressions. This model leverages the global context for symbol segmentation, symbol recognition, and relation classification [5]. The method imposes a constraint to ensure the model outputs the relation classification at a precise time step, simultaneously addressing both symbol segmentation and relation classification. As shown in Level 0 of Fig. 2, the model takes the raw input sequence of strokes and segments them into symbols (e.g., 'x', '2', '+', 'a'). A key characteristic of this stage is that it primarily identifies relations between symbols that are consecutive in the writing order. We currently restrict the BLSTM's relation predictions to consecutive-in-writing-order pairs to exploit temporal context. However, writing order can vary (e.g., users draw subscripts out of sequence). In practice this can cause missing edges for non-adjacent symbols. Thus, we resolve the problem by feeding the BLSTM arbitrary symbol pairs, which discussed in the next step of symbol-level graph construction.

Symbol-Level Graph Construction The relations identified by the BLSTM are insufficient for capturing the full 2D structure. Therefore, in the second stage, we use a parser based on a 2D Stochastic Context-Free Grammar (2D-CFG) [17] and the Cocke-Younger-Kasami (CYK) algorithm to build a more complete primitive graph. As depicted in Level 1 of Fig. 2, this parser considers all pairs of recognized symbols, not just consecutive ones. To determine the spatial relation between any two non-consecutive symbols (e.g., 'x' and 'a'), a new, temporary input sequence is constructed containing only the strokes for those two symbols. This new sequence is then fed into the same pre-trained BLSTM model to obtain a predicted spatial relation and its probability. This ensures all plausible pairwise relations are considered.

Filter redundant edges using Line-Of-Sight We also applied the LOS algorithm [1] for filtering redundant edges. Let E_{CYK} be the set of edges proposed by CYK algorithm, E_{LOS} be the set of edges proposed by LOS algorithm. The output set of edges is:

$$E_{\text{CYK} \ \& \ \text{LOS}} = E_{\text{CYK}} \wedge E_{\text{LOS}} \quad (1)$$

The result of this entire process is a dense, symbol-level graph, like the one shown in the "Symbol Graph" part of 1, which serves as the input to our GNN.

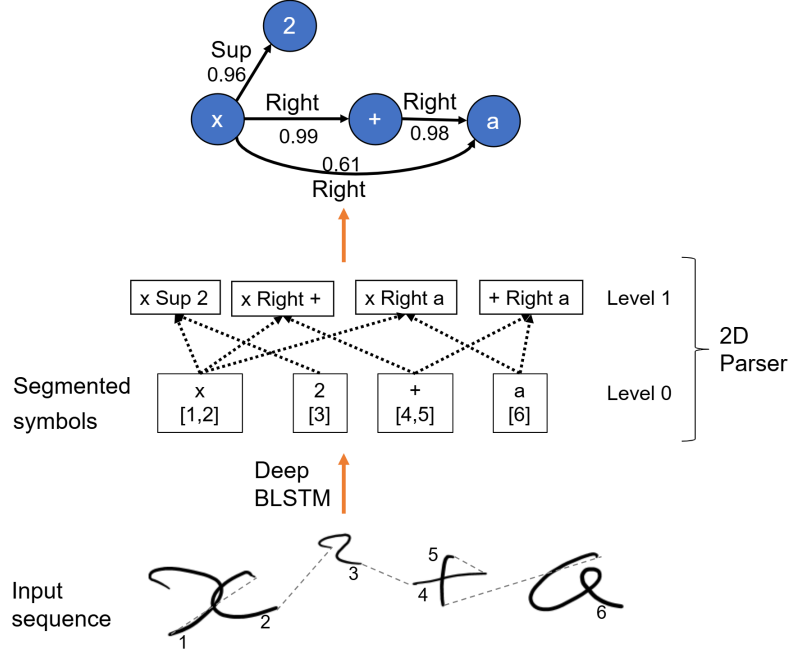


Fig. 2. Symbol-level graph constructed by deep BLSTM and 2D CYK parsing.

3.3 Edge Graph Attention Networks for Link Prediction

To further refine the symbol-level graphs generated by the CYK parsing and Line-of-Sight filtering, we utilize Edge Graph Attention Networks (EGAT). The EGATs are specifically designed to handle graph data by capturing rich structural relationships through attention mechanisms applied directly to edge features and node features. This capability makes EGAT particularly suitable for tasks involving intricate spatial relationships, such as handwritten mathematical expression recognition. Figure 3 illustrates the EGAT that refines the initial symbol-level graph to form an SLT output.

EGAT Architecture Our EGAT model is implemented using the Deep Graph Library (DGL) on a PyTorch backend. It comprises four stacked EGATConv layers [3], each of which updates node and edge embeddings via an edge-integrated attention mechanism. The final embeddings are passed to a two-layer MLP classification head, which outputs logits over the relation classes. All layers use LeakyReLU activations and 0.2 dropout to mitigate overfitting. The attention mechanism in EGAT is defined by:

$$e_{ij} = F_{\text{att}}(f'_{ij}) \quad (2)$$

$$f'_{ij} = \text{LeakyReLU}(A[h_i || f_{ij} || h_j]) \quad (3)$$

where f'_{ij} is the updated edge features, h_i and h_j represent the embedding vectors of nodes i and j respectively, f_{ij} denotes the original edge features connecting nodes i and j . A is a learnable weight matrix that projects the concatenated features into a new embedding space suitable for attention computation. The operator $||$ indicates concatenation of two feature vectors.

Input Features Our EGAT leverages both node and edge information:

1. **Node Features** Each node’s feature is a one-hot vector encoding its recognized symbol class.
2. **Edge Features** For each candidate edge, we take its upstream predicted relation (one-hot) and multiply it by the scalar confidence score output by the BLSTM. This produces a weighted edge embedding that encodes both the relation type and its confidence.

Link Prediction After propagation through the EGAT layers, a final MLP classifier predicts whether an edge should be included or not in the output of SLT. The prediction for each edge uses a combined feature vector, which we term the NodeEdgeFeature. It is created by concatenating the final embeddings of the source node, the edge itself, and the destination node:

$$\text{NodeEdgeFeature} = [h_i || f'_{ij} || h_j] \quad (4)$$

This combined feature is then passed to the MLP classifier. This ensures the final classification is conditioned on the full context of the edge and the symbols it connects. Instead of NodeEdgeFeature, the link can also be predicted from the EdgeFeature of f'_{ij} .

Training The loss of link prediction L is computed using Binary Cross Entropy loss. Training involves optimizing this loss across the entire symbol-level graph dataset, adjusting network parameters to minimize prediction errors. The Binary Cross-Entropy Loss:

$$L = \sum_{i=1}^N y_i \cdot \log p_i + (1 - y_i) \cdot \log(1 - p_i) \quad (5)$$

where y_i is the ground-truth label indicating whether an edge exists or not, and p_i is the predicted probability, N is the number of dataset samples.

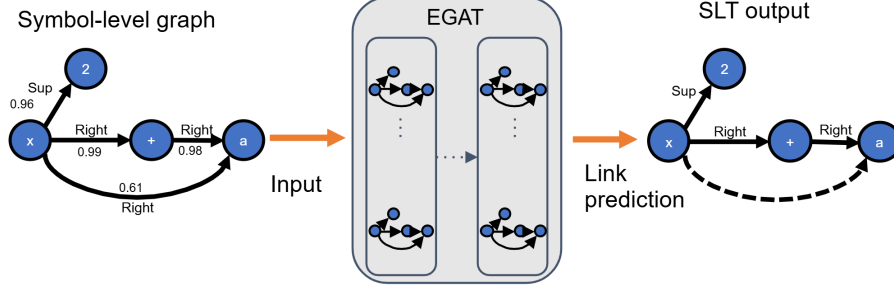


Fig. 3. The link prediction EGAT refines the initial symbol-level graph, removing edges to form an SLT.

4 Experiments

4.1 Datasets and metrics

We evaluated our model using the CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions) dataset. We used the train dataset from CROHME2019 and evaluated it on three test sets: CROHME2014, CROHME2016, and CROHME2019. The train dataset is used to train an LSTM model for symbol segmentation, symbol recognition, and relation classification, and used to train the EGAT model for link prediction.

For the deep BLSTM model, we used a stack of three BLSTM layers, each containing two LSTM layers (bidirectional) with 128 hidden units. The input sequence of coordinates in a stroke was sampled by the Ramer method, then extracted four features: the sine and cosine of the writing direction, the normalized distance between two consecutive points and the pen-up/pen-down binary value. For the GNN, we use the stack of four EGAT layers followed by a fully connected layer for link prediction.

We measured the performance of symbol segmentation, symbol recognition, and relation classification using the BLSTM model. For symbol-level graph construction, we evaluated the coverage and redundancy of edges. Coverage measures the recall of ground-truth edges in the primitive graph:

$$\text{Coverage} = \frac{|\{e | e \in G_{\text{primitive}} \wedge e \in G_{\text{ground-truth}}\}|}{|\{e | e \in G_{\text{ground-truth}}\}|} \times 100\% \quad (6)$$

Redundancy measures the percentage of edges rather than that of ground-truth edges in the primitive graph:

$$\text{Redundant} = \left(\frac{|\{e | e \in G_{\text{primitive}}\}|}{|\{e | e \in G_{\text{ground-truth}}\}|} - 1 \right) \times 100\% \quad (7)$$

For link prediction, we evaluated link prediction accuracy. At the expression level, we evaluated the expression rate, the accuracy of recognizing entire mathematical expressions, and the structure rate, whether all the spatial relationships in a mathematical expression are correct.

4.2 Experiments and results

In the first evaluation, we evaluated the primitive graph produced by the LSTM model and the CYK parser. The result is shown in Table 1. We also evaluated the graph produced by the LOS algorithm and that by the CYK parser filtered by the LOS graph.

Table 1. Primitive graph evaluation on CROHME 2014 dataset

Method	Seg. (%)	Seg. + Cls. (%)	Rel. (%)	Cov. (%)	Red. (%)
BLSTM-CYK	98.09 (87.84)	92.53 (60.37)	92.99 (72.01)	94.69 (79.16)	107.65
BLSTM-LOS	98.09 (87.84)	92.53 (60.37)	92.64 (69.97)	96.09 (84.78)	722.30
BLSTM CYK & LOS	98.09 (87.84)	92.53 (60.37)	92.64 (69.97)	94.33 (76.92)	87.74

The value in parentheses is measured at expression level.

Abbreviation: Seg. (Segmentation), Cls. (Classification), Rel. (Relation) , Cov. (Coverage), Red. (Redundant)

Furthermore, the values in parentheses denote the evaluation metrics at the expression level. An expression is considered correctly recognized only if both its symbols and relations are accurately classified. As a result, these metrics set an upper bound on the overall expression-level performance.

Our analysis indicates that the LOS algorithm effectively recovers missing relationships but significantly elevates graph complexity due to redundant edges. Applying LOS as a filtering step with CYK reduces redundancy substantially with minimal sacrifice in coverage, suggesting an effective compromise between coverage and complexity.

Table 2 presents the link prediction accuracies on the CROHME 2014, 2016, and 2019 test sets. The LOS algorithm achieves superior link prediction performance, with accuracies of 97.97%, 97.99%, and 98.12% for each respective dataset. However, its high number of redundant edges adversely impacts the expression-level accuracy compared to alternative approaches. Notably, when CYK parsing is integrated with LOS filtering, there is a consistent improvement across all datasets with link accuracies of 92.61%, 91.61%, and 92.78%, demonstrating the robust performance of this combined approach in balancing link prediction with overall expression-level reliability.

Table 3 illustrates the performance on full mathematical expression recognition across the three test sets. Our integrated CYK & LOS approach achieved consistently higher Expression Rates (37.59%, 38.35%, 38.86%) and Structure Recognition Rates (51.07%, 48.60%, 51.42%) compared to individual CYK or

Table 2. EGAT link prediction evaluation

Input graph	Test 2014	Test 2016	Test 2019
	Link acc. (%)	Link acc. (%)	Link acc. (%)
BLSTM-CYK	92.21 (57.00)	91.49 (56.22)	92.21 (53.97)
BLSTM-LOS	97.97 (53.22)	97.99 (52.67)	98.12 (54.73)
BLSTM-CYK & LOS	92.61 (60.06)	91.61 (57.09)	92.78 (60.22)

The value in parentheses is measured at expression level

LOS methods. These results confirm the effectiveness of combining structural parsing and redundancy filtering.

Table 3. Math expression evaluation

Input graph	Test 2014		Test 2016		Test 2019	
	Exp. (%)	Struct. (%)	Exp. (%)	Struct. (%)	Exp. (%)	Struct. (%)
BLSTM-CYK	36.36	49.23	38.18	48.42	36.07	47.28
BLSTM-LOS	34.83	47.09	37.01	46.11	37.66	48.62
BLSTM-CYK & LOS	37.59	51.07	38.35	48.60	38.86	51.42

In the ablation study summarized in Table 4, we evaluated the impact of including NodeEdgeFeatures in our EGAT model. This inclusion notably improved link accuracy, validating the critical role of comprehensive feature representation in GNN-based parsing methods.

Table 4. Ablation study of EGAT network

Method	Test 2014	Test 2016	Test 2019
	Link acc. (%)	Link acc. (%)	Link acc. (%)
EGAT + EdgeFeature	91.50 (55.16)	91.11 (51.23)	91.82 (55.11)
EGAT + NodeEdgeFeature	92.61 (60.06)	91.61 (57.09)	92.78 (60.22)

Finally, Table 5 compares our GNN-based method with state-of-the-art models across multiple test sets. Although our approach does not surpass the top-performing systems such as MyScript or Samsung, it demonstrates competitive Expression rate and Structure Recognition Rates. Our method outperforms the graph-based method for building SLT such as the MST algorithm [2], tree BLSTM [13] by a large margin. Compare to other GNN approaches which used GNN output features for symbol and relation classification, we are behind the expression rate since our method is limited by symbol classification performance at expression-level.

Our method demonstrates competitive performance, particularly when compared to other explicit graph-based methods like MST [2], which it outper-

Table 5. Comparison to state-of-the-art models on CROHME datasets

System	Type	Test 2014		Test 2016		Test 2019	
		Exp. (%)	Struct. (%)	Exp. (%)	Struct. (%)	Exp. (%)	Struct. (%)
MyScript	Gram	62.68	-	67.65	88.14	79.15	90.66
UPV/Wiris	Gram	37.22	-	49.61	74.28	60.13	79.15
Tokyo	Gram	25.66	-	43.94	61.55	39.95	58.22
Samsung	Gram	-	-	65.76	-	79.82	89.32
MST	Graph	26.88	-	-	-	-	-
Tree-BLSTM	Graph	29.91	-	27.03	-	-	-
Tree-construct.	Graph	44.12	58.62	41.76	49.43	-	-
TAP	EncDec	46.90	-	44.80	-	-	-
iFlyTek	EncDec	-	-	57.02	-	80.73	91.49
G2G	GNN	-	-	52.05	-	-	-
LGM-EGAT	GNN	-	-	56.41	-	58.22	83.07
GGM-EGAT	GNN	-	-	56.67	-	60.72	83.74
Our	GNN	37.59	51.07	38.35	48.60	38.86	51.42

forms by a large margin. While our expression rates currently lag behind top-performing grammar-based (MyScript) and end-to-end encoder-decoder systems (iFlyTek), this highlights a key trade-off. Our pipeline approach separates symbol recognition from structural analysis, meaning errors from the initial BLSTM recognizer can propagate and limit the final expression rate. However, the strength of our approach is its explicit structural output (a graph), which is more interpretable and editable than the LaTeX strings from encoder-decoder models. The strong Structure Rate (e.g., 51.42% on CROHME 2019) shows that the GNN is highly effective at its specific task: learning and predicting the correct graph structure. This indicates significant potential, and future work focused on improving the initial symbol recognizer could close the gap with end-to-end systems. Furthermore, when comparing with other GNN-based methods like LGM-EGAT [9], our performance is comparable, indicating that the current state of symbol-level GNNs for this task is still evolving. The performance gap suggests that future research should focus on more robust primitive graph construction and more powerful GNN architectures.

5 Conclusion

In this paper, we presented a novel framework for handwritten mathematical expression recognition that effectively captures structural relationships using symbol-level graph representations. Our approach addresses key limitations of existing stroke-level graph recognition methods by reducing the graph complexity. Integrating a BLSTM network for simultaneous segmentation, recognition, and spatial relation classification provided robust accuracy on three tasks. The subsequent use of a CYK parsing algorithm, coupled with a practical Line-of-Sight heuristic, generated a clear and concise initial graph structure. Finally,

our customized EGAT architecture accurately predicted and refined spatial relationships, producing Symbol Layout Tree as output. Experimental evaluations on the CROHME benchmark dataset demonstrated our proposed symbol-level graph approach’s competitive performance in recognition accuracy.

Future research directions include exploring improvements on symbol-level graph construction by analyzing the failure cases, improving link prediction with graph data augmentation and further optimizing computational efficiency for real-time applications.

References

1. Hu, L., Zanibbi, R.: Line-of-sight stroke graphs and parzen shape context features for handwritten math formula representation and symbol segmentation. *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR* **0**, 180–186 (7 2016). <https://doi.org/10.1109/ICFHR.2016.0044>
2. Hu, L., Zanibbi, R.: Mst-based visual parsing of online handwritten mathematical expressions. In: *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*. vol. 0, pp. 337–342. Institute of Electrical and Electronics Engineers Inc. (7 2016). <https://doi.org/10.1109/ICFHR.2016.0070>
3. Kamiński, K., Ludwiczak, J., Jasiński, M., Bukala, A., Madaj, R., Szczepaniak, K., Dunin-Horkawicz, S.: Rossmann-toolbox: a deep learning-based protocol for the prediction and design of cofactor specificity in rossmann fold proteins. *Briefings in bioinformatics* **23** (1 2022). <https://doi.org/10.1093/BIB/BBAB371>
4. Le, A.D., Nakagawa, M.: A system for recognizing online handwritten mathematical expressions by using improved structural analysis. *International Journal on Document Analysis and Recognition* **19**, 305–319 (12 2016). <https://doi.org/10.1007/s10032-016-0272-4>
5. Nguyen, C.T., Truong, T.N., Nguyen, H.T., Nakagawa, M.: Global context for improving recognition of online handwritten mathematical expressions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **12822 LNCS**, 617–631 (2021). https://doi.org/10.1007/978-3-030-86331-9_40
6. Tang, J.M., Wu, J.W., Yin, F., Huang, L.L.: Offline handwritten mathematical expression recognition via graph reasoning network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **13188 LNCS**, 17–31 (2022). https://doi.org/10.1007/978-3-031-02375-0_2
7. Truong, T.N., Nguyen, H.T., Nguyen, C.T., Nakagawa, M.: Learning symbol relation tree for online handwritten mathematical expression recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **13189 LNCS**, 307–321 (2022). https://doi.org/10.1007/978-3-031-02444-3_23
8. Wu, J.W., Yin, F., Zhang, Y.M., Zhang, X.Y., Liu, C.L.: Graph-to-graph: Towards accurate and interpretable online handwritten mathematical expression recognition. *Proceedings of the AAAI Conference on Artificial Intelligence* **35**, 2925–2933 (5 2021). <https://doi.org/10.1609/AAAI.V35I4.16399>
9. Xie, Y., Zanibbi, R., Mouchère, H.: Local and global graph modeling with edge-weighted graph attention network for handwritten mathematical expression recognition. *arXiv preprint arXiv:2410.18555* (10 2024)

10. Yamamoto, R., Sako, S., Nishimoto, T., Sagayama, S.: 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 (10 2006)
11. Zhang, J., Du, J., Dai, L.: Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia* **21**, 221–233 (1 2019). <https://doi.org/10.1109/TMM.2018.2844689>
12. 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 (11 2017). <https://doi.org/10.1016/J.PATCOG.2017.06.017>
13. Zhang, T., Mouchère, H., Viard-Gaudin, C.: A tree-blstm-based recognition system for online handwritten mathematical expressions. *Neural Computing and Applications* **32**, 4689–4708 (5 2020). <https://doi.org/10.1007/s00521-018-3817-2>
14. Zhao, W., Gao, L.: Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **13688 LNCS**, 392–408 (2022). https://doi.org/10.1007/978-3-031-19815-1_23
15. Zhao, W., Gao, L., Yan, Z., Peng, S., Du, L., Zhang, Z.: Handwritten mathematical expression recognition with bidirectionally trained transformer. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **12822 LNCS**, 570–584 (2021). https://doi.org/10.1007/978-3-030-86331-9_37
16. Zhu, J., Gao, L., Zhao, W.: Ical: Implicit character-aided learning for enhanced handwritten mathematical expression recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* pp. 21–37 (2024). https://doi.org/10.1007/978-3-031-70549-6_2
17. Álvaro, 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). <https://doi.org/10.1016/J.PATREC.2012.09.023>