Line-of-Sight with Graph Attention Parser (LGAP) for Math Formulas

Ayush Kumar Shah(✉) and Richard Zanibbi

Document and Pattern Recognition Lab, Rochester Institute of Technology, Rochester, NY 14623, USA
{as1211,rxzvcs}@rit.edu

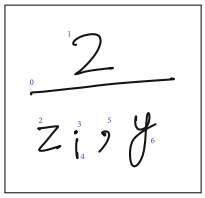
Abstract. Recently there have been notable advancements in encoder-decoder models for parsing the visual appearance of mathematical formulas. These approaches transform input formula images or handwritten stroke sequences into output strings (e.g., LaTeX) representing recognized symbols and their spatial arrangement on writing lines (i.e., a Symbol Layout Tree (SLT)). These sequential encoder-decoder models produce state-of-the-art results but suffer from a lack of interpretability: there is no direct mapping between image regions or handwritten strokes and detected symbols and relationships. In this paper, we present the Line-of-sight with Graph Attention Parser (LGAP), a visual parsing model that treats recognizing formula appearance as a graph search problem. LGAP produces an output SLT from a Maximum Spanning Tree (MST) over input primitives (e.g., connected components in images, or handwritten strokes). LGAP improves the earlier QD-GGA MST-based parser by representing punctuation relationships more consistently in ground truth, using additional context from line-of-sight graph neighbors in visual features, and pooling convolutional features using spatial pyramidal pooling rather than single-region average pooling. These changes improve accuracy while preserving the interpretibility of MST-based visual parsing.

Keywords: math formula recognition · attention masks · spatial pyramidal pooling · line-of-sight neighbors · graph search

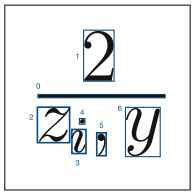
1 Introduction

Mathematical notations are widely used in technical documents to represent complex relationships and concepts concisely. However, most retrieval systems only accept text queries, and cannot handle structures like mathematical formulas. To aid in the development of search engines that *can* process queries containing mathematical notation [3,8,22,25,42,45], recognition of mathematical expressions is needed for indexing formulas in documents, and to support handwritten input of formulas in queries [21,23,28,50]. These documents include both born-digital PDF documents (e.g., where symbols in a formula may be available, but not formula locations [5,29]), and scanned documents, for which OCR

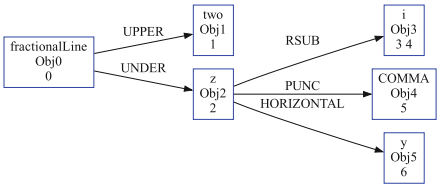
G. A. Fink et al. (Eds.): ICDAR 2023, LNCS 14191, pp. 401–419, 2023.
https://doi.org/10.1007/978-3-031-41734-4_25



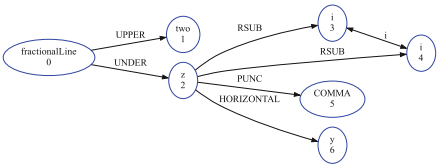
(a) Online handwritten formula



(b) Typeset formula (rendered LaTeX)



(c) Symbols: Symbol Layout Tree (SLT)



(d) Strokes/CCs: Labeled graph

Fig. 1. Formula structure at the level of symbols (c) and input strokes and connected components (d) for a formula both online handwritten (a) and rendered using LaTeX (b). Numeric identifiers for strokes (a) and their corresponding connected components (b) are shown in blue. Note that the 'i' comprised of strokes/CCs {3,4} is shown in one node in (c), but two nodes in (d). (Color figure online)

results for formulas can be unreliable. We focus here on the task of parsing the visual structure of formulas from images and handwritten strokes after formulas have been detected/isolated, and parsing isolated formulas from connected components in binary images in particular (see Fig. 1).

Parsing mathematical formulas belongs to a broader class of graphical structure recognition tasks that includes visual scene understanding, road type classification, molecular analysis, and others. The Line-of-Sight with Graph Attention Parser (LGAP) presented in this paper is a refinement of the earlier Query-Driven Global Graph Attention formula parser (QD-GGA [20]). Both systems produce *Symbol Layout Trees* (SLTs [43], see Fig. 1c) representing the hierarchical arrangement of symbols on writing lines, and the nesting of writing lines around symbols (e.g., for superscripts and subscripts).

In recent years, there have been several approaches to math formula recognition, including pixel-based encoder-decoder methods that produce state-of-the-art results, but are slow and lack interpretability; syntax-based methods that use grammar-based rules, which are less robust and computationally expensive; and graph search methods such as QD-GGA that are fast and interpretable, but have so far been unable to match the accuracy of encoder-decoder models, in part due to the limited use of context and attention.

In this paper, we describe the LGAP system, which improves the visual features and use of context in QD-GGA. This is achieved by incorporating line-of-sight (LOS) neighbors to capture additional local context [13,14], reducing

the loss of spatial information through pyramidal pooling [11], and modifying the representation of punctuation relationships to avoid requiring edges that are missing in many LOS graphs. Our main research questions are:

1. Can recognition accuracy of the parser be increased by modifying the punctuation representation in ground truth?
2. Will increasing use of context in image features by adding LOS graph neighbors for primitives (nodes) and primitive pairs (edges) increase accuracy?
3. Will spatial pyramidal pooling of (SPP) convolutional features improve accuracy over the single window average pooling used in QD-GGA?

2 Related Work

For parsing of mathematical expressions, syntactic (grammar-based) approaches, encoder-decoder approaches converting pixel or strokes to strings, and graph search (tree-based) approaches are commonly used. Chan et al. [6], Zanibbi et al. [43], Zhelezniakov et al. [49], and Sakshi et al. [27] have surveyed a wide range of math recognition systems over the past few decades. The recognition of math expressions dates back to 1967 when Anderson introduced a syntax-directed approach for recognition of two-dimensional handwritten formulas including arithmetic expressions and matrices, using a 2D top-down parsing algorithm employing an attribute grammar [4]. Many systems followed the syntactic approach using a context-free grammar (CFG) of some form, including Stochastic CFGs [2] and a parser to output the parse tree and a LaTeX string [2,5,24,33]. However, there are challenges with syntactic approaches, in that the formula symbols and structures often need to be redefined [41] and it is difficult to design a universal grammar for all notation variants [48], making the syntactic approaches less robust.

In this section, we discuss the contributions and limitations of encoder-decoder models, graph-search based models, and models using graph attention and context, and briefly describe the similarities and differences in our models.

Encoder-decoder models are deep neural network architectures in which an encoder produces a feature embedding for input data in a lower-dimensional space, while the decoder converts the embedded input representation back into the original input, or into another representation. Generally, a Convolution Neural Network (CNN) is used for encoding pixel-level image features, and Recurrent Neural Networks (RNNs) are used as decoders to produce a string representation of a formula (e.g., in LaTeX) with some form of attention mechanism. Encoder-decoder models are used in other sequence prediction problems such as image caption generation [39], speech recognition, and scene text recognition [40], and have produced state-of-the-art results for these problems as well as math formula recognition [32,36,46,47]. The end-to-end Track, Attend and Parse (TAP) system [46] is a popular example. TAP obtained state-of-the-art results for many years, mostly because of the intelligent use of ensemble models that combine online and offline features, as well as different types of attention mechanisms.

Wu at al. [36] aim to improve generalization and interpretability in sequence-based encoder-decoder models by utilizing a tree-based decoder with attention, removing the need for the spatial relationships to be in strict order. A few other variations include a counting aware network [18] using symbol counting as a global symbol-level position information to improve attention, and a stroke constrained network [35] which use strokes as input primitives for their encoder-decoder, to improve the alignment between strokes and symbols.

A limitation of encoder-decoder models is the interpretibility of their results. Error diagnosis in these models is challenging due to the lack of a direct correspondence between the input (image regions or strokes) and the symbols and relationships produced in an output LaTeX string or SLT: the exact correspondence between input primitives (pixels, CCs, or strokes) and the output is unknown. Hence, there is no way to identify errors at the input primitive level. They also tend to be computationally expensive: the encoder-decoder parsers use an RNN decoder with attention-based mechanisms, which requires sequential processing and greater memory usage due to maintaining hidden states across the entire input sequence.

Graph-Based Methods and MST-Based Parsing. Representing mathematical formulas as trees is more natural than images or one-dimensional strings. For example, LaTeX expresses formulas as a type of SLT with font annotations. Mathematical expression recognition can alternatively be posed as filtering a graph to produce a maximum score or minimum cost spanning tree (MST) representing symbols and their associated spatial relationships. Eto et al. [10] were the first to take this approach, creating a graph with symbol nodes containing alternative labels, and candidate spatial relationships on edges with associated costs. Formula structure was obtained from extracting a minimum spanning tree amongst the relationship edges, along with minimizing a second measure of cost for the global formula structure.

For graph-based parsing, an input graph defined by a complete graph connecting all primitives is natural. However, this leads to lots of computation, and makes statistical learning tasks challenging due to input spaces with high variance in features: this motivates reducing variance through strategic pruning of input graph edges. Line-of-sight (LOS) graphs select edges where strokes in a handwritten formula or connected components in an image are mutually visible [14,19]. Systems such as Hu et al. [13], LPGA [19], and QD-GGA [20] use this LOS input graph constraint, and select the final interpretation as a *directed* Maximum Spanning Tree (MST). QD-GGA [20] extracts formula structure using an MST over detected symbols, extending previous approaches [19,30] by adding a multi-task learning (MTL) framework, and graph-based attention used to define visible primitives in images used to generate visual features for classification. QD-GGA uses Edmond's arborescence algorithm [9] to obtain a directed Maximum Spanning Tree (MST) between detected symbols, maximizing the sum of spatial relationship classification probabilities obtained from an end-to-end CNN network with attention. Using directed MSTs allows many invalid interpretations to be pruned, as the output graph is a rooted directed tree (as SLTs are).

Graph-based parsing is more natural for mathematical expressions. Unlike encoder-decoder models, the mapping between the input and the output can be obtained from labels assigned in the output to the nodes (strokes or connected components) and edges provided in the input [44]. Error metrics at the symbol and primitive levels for segmentation, symbol classification, and relation classification can be computed directly from the labeled output graphs. Also, like encoder-decoder models, these techniques do not require expression grammars, requiring only a vocabulary of symbol and relationship types. However, these models, although fast with easier interpretability, have not been able to match the accuracy of encoder-decoder models. Possible reasons may include a lack of global context, attention, and spatial information in the current models.

The use of graph neural networks with attention has been seeing more use in math parsing to capture context between primitives using graph edges directly. For instance, Peng et al. [26] use a gated graph neural network (GGNN) in the encoder stage as a message passing model to encode CNN features with visual relationships in the LOS graph. Wu et al. [37] use GNN-GNN encoder-decoder (modified Graph Attention Network [34] encoder, modified Graph Convolution Network [16] decoder) to utilize graph context. Tang et al. [32] aims to learn structural relationships by aggregating node and edge features using a Graph Attention Network to produce SLTs by simultaneous node and edge classification, instead of the sequence representation used for encoder-decoder models. They produce the output SLT by filtering 'Background' nodes and 'No-Relation' edges. Note the contrast with the use of Edmonds' algorithm in QD-GGA and LGAP, where all primitives are assumed to belong to a valid symbol, and an MST algorithm selects directed edges between symbols obtained *after* merging primitives predicted to belong to the same symbol to produce an SLT.

This Paper. LGAP is an MST-based parser for math formulas. In LGAP, we focus specially on improving QD-GGA features by intelligent use of context and improved attention mechanisms. In our approach, we employ LOS not only to construct the input graph, but also to incorporate LOS neighbors of primitives as supplementary local context for visual features. Additionally, we adopt spatial pyramidal pooling [11] rather than the isolated average pooling used in QD-GGA to mitigate a loss of spatial information. Likewise, we deal with issues in the LOS graph representation for punctuation. Punctuation was being represented by special 'PUNC' edges between a symbol and its adjacent '.' or ',' when present, but often a symbol blocks the line-of-sight between the parent symbol at left and a punctuation symbol at right.

3 Recognition Task: Punctuation Representation and Metrics for Symbols, Relationships, and Formulas

In this paper we explore improvements in MST-based parsing, focusing on typeset formulas for our initial investigation. Typeset formulas are more visually consistent than handwritten formulas, with lower variance in visual features,

and we chose to examine whether our modifications are effective in this simpler setting first. We will consider handwritten formulas at a later time.

In this Section, we define our formula recognition task in terms of inputs (isolated formula CCs), outputs (SLTs), and the evaluation metrics used to quantify performance. In particular, we have modified our task by changing the representation of spatial relationships for punctuation symbols: appearance-wise the position of punctuation relative to its associated symbol at left is more similar to a subscript than horizontal adjacency (e.g., A_1 vs. AB). The use of LOS graphs in LGAP poses some additional challenges, which we address below.

Dataset. We use InftyMCCDB-2[1], a modified version of InftyCDB-2 [31]. The dataset contains binary images for isolated formulas from scanned articles, with formulas containing matrices and grids removed. The training set has 12,551 formulas, and the test set contains 6830 formulas. The dataset includes 213 symbol classes, which may be reduced to 207 by merging visually similar classes (e.g., *ldots* and *cdots*, *minus* and *fractional line*), and 9 relationship classes: *Horizontal*, *Rsub* (right subscript), *Rsup* (right superscript), *Lsub* (left subscript), *Lsup* (left superscript), *Upper*, *Under*, *PUNC* (for punctuation), and *NoRelation* (see Figs. 1d and 2). The training and test sets have approximately the same distribution of symbol classes and relation classes.

Evalation Metrics. We report expression recognition rates for (1) *Structure:* unlabeled SLTs with correct nodes (CC groups for symbols) and edges (for relationships), and (2) labeled SLTs where symbol classes and relationship types must also be correct. For a finer-grained analysis, we also report detection and classification F-scores for symbols and relationships using the LgEval library originally developed for the CROHME handwritten formula recognition competitions [23]. F-scores are the harmonic mean of recall and precision for the detection of target symbols and relationships (2RP/(R+P)). We report both: (1) *detection f-scores*: quantify properly detected/segmented CC groups for symbols, and the presence of an edge (relationship) between two properly segmented symbols, and (2) detection + classification f-scores: here a symbol or relationship is correct if has been detected correctly *and* assigned the correct symbol or relationship label.

Modified Punctuation Relationship in Ground Truth SLTs. Mahdavi et al. [19] added a new spatial relationship called *PUNC* in the InftyMCCDB-2 dataset for distinguishing horizontal relationships with punctuation symbols from other horizontal relationships to make their visual features more consistent and thereby identifiable. However, this modification can lead to missing ground truth punctuation edges in the input LOS graph, often when a symbol has a subscript and is followed by a punctuation symbol on its writing line. This happens since there is good chance that the line of sight between the parent symbol and the punctuation is blocked by one or more primitives between them, usually a large subscript. For the example in Fig. 2, the ground truth edge between the symbols 'z' and 'COMMA' is blocked by the primitives corresponding to 'i'. Here

[1] https://www.cs.rit.edu/~dprl/data/InftyMCCDB-2.zip.

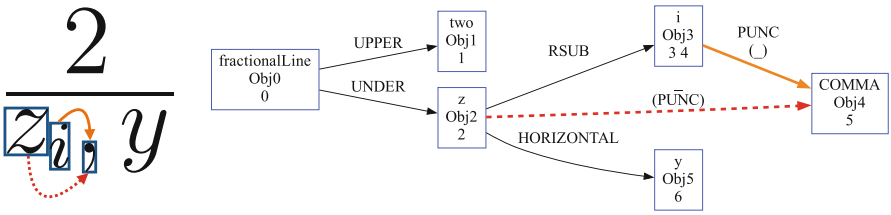


Fig. 2. Modified puncutation (*PUNC*) ground truth representation. The old *PUNC* edge is shown using red dashed arrows, and its corresponding new edge is shown with solid orange arrows. The original *PUNC* edge between nodes 'z' and 'COMMA' is missing in the LOS graph, as can be seen in Fig. 1a and 1b (Color figure online).

the *PUNC* relationship will be missed, due to the missing edge in the input LOS graph representation. Additional heuristics in LPGA [19] modify the LOS graph to add missing punctuation based on angles and relative sizes of symbols, which are not very robust.

To address this issue, we modified the assigned parent node for *PUNC* edges as shown in Fig. 2. Instead of the original parent, which may be far left on the writing line and/or blocked in the LOS graph, we assign the PUNC parent node to the symbol immediately at left of the punctuation symbol. This modification ensures all *PUNC* edges are included in input LOS graphs. This also improves training/learning, as relationships between *PUNC* parent and child nodes using more consistent visual features (see Sect. 5).

4 Line-of-Sight with Graph Attention Parser (LGAP)

In this section we describe the MST-based LGAP parsing model inputs and outputs, its multi-task end-to-end CNN classification model, and finally the steps to select symbols and extract an MST to output a symbol layout tree. LGAP was implemented in PyTorch with Python NetworkX library for representing graphs and running Edmonds' algorithm to obtain MSTs/SLTs.[2]

4.1 Inputs and Outputs

Query and LOS Context Attention Masks. For formula images, we extract CC contours by first smoothing them and then sampling contours to produce trace points. For handwritten formula images, trace points are already available. These trace points are interpolated, normalized, and scaled to a height of 64 pixels while preserving the aspect ratio: this results in formula images with a fixed height but varying width. The full normalized formula image is fed into a CNN encoder backbone to compute the global visual features, and points on sampled contours are used to construct the LOS input graph (described below).

[2] LGAP open source implementation: https://gitlab.com/dprl/qdgga-parser.

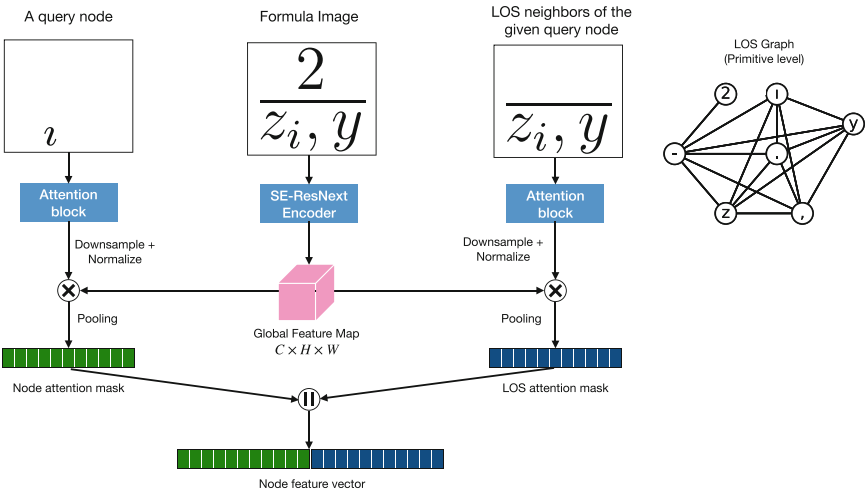


Fig. 3. Binary attention masks in LGAP. The input primitive query mask (represented here by the base of the letter 'i') and its corresponding LOS masks are used to generate the attention masks by performing element-wise multiplication with the global feature map. The two attention masks are concatenated to get the node feature vector that is utilized for symbol classification. Note that the same process is applied to the primitive pair binary masks and LOS mask to generate the primitive pair feature vector for classifying directed edges.

Input regions are processed to produce binary attention masks created for input *queries* corresponding to individual primitives (nodes) or primitive pairs sharing an edge in the LOS graph. Binary masks for directed edges between primitive pairs are concatenated with the binary mask for the parent primitive, to increase identifiability and thus classification accuracy [20]. Query binary masks filter the CNN global feature map, providing focus on image regions pertinent for three classification tasks: (1) masking individual CC/strokes for symbol classification (for nodes), and masking CC/stroke pairs for (2) segmentation and (3) spatial relationship classification (for directed edges).

In LGAP, we introduce an additional binary *LOS mask*, which consists of a binary sum (OR) of the query mask with the binary mask of *all* neighboring primitives in the LOS graph (see Fig. 3). These additional LOS binary masks serves to provide visual spatial context information from the neighbors in the LOS graph. As for primitive and primitive pair binary masks, LOS masks are downsampled and weight the global feature map produced by the encoder in order to focus on the relevant parts of the input image for the three classification tasks (symbols, segments, relationships).

Primitive LOS Graph. Our parser uses as input either a set of online handwritten strokes, or a set of connected components (CCs) from a typeset formula image (see Fig. 1a, 1b). A handwritten stroke is represented by a list of 2D (x,y) coordinates obtained from sampled positions of a pen, trackpad, or other touch

device. When parsing online handwritten formulas, our input is a graph over strokes images: strokes are individually rendered using their stroke coordinates. To parse typeset formulas, we construct a graph over connected components. For both handwritten and typeset formulas, we assume that *primitives* (stroke images or CCs) belong to exactly one symbol.

Initially, the input is a complete graph with an edge between every pair of primitives (CCs or strokes). As seen in Fig. 4, we then select edges in the Line of Sight (LOS) graph [7] as suggested by Hu et al. [14]. In the LOS graph, edges exist only for primitive pairs where an unobstructed line can be drawn from the center of one primitive to a point on the convex hull of the contour for the other [19]. This reduces the space of output graphs and number of classifications needed, with few deletions of pertinent edges [14]. In Fig. 4, all edges between the '2' and other CCs other than the fraction line are pruned.

Output: Symbol Layout Tree. Formula appearance is commonly represented as a tree of symbols on writing lines known as a Symbol Layout Tree (SLT, see Fig. 1c [43]). SLTs have popular encodings such as LaTeX and Presentation MathML that include additional formatting information such as fonts, font styles (e.g., italic), and spacing. Because they describe spatial arrangements of complete symbols, SLTs are identical for formulas in handwritten strokes and typeset images. However, the number and arrangement of strokes or CCs may differ for a single formula. For the example in Fig. 1, the primitive label graphs for the image and handwritten version are identical so that CCs and strokes have an exact correspondence. This is not always the case, for example when an 'x' drawn with two handwritten strokes appears as a single CC in a typeset image.

In LGAP we produce an SLT over symbols from our directed LOS input graph over strokes or CCs. Figure 4 shows how a complete graph over primitives is constrained and annotated with classification probabilities in stages, producing an SLT as output. [44]. An example of a labeled LOS graph representing an SLT is shown in Fig. 1d for both handwritten strokes and image CCs. There are bidirectional edges between strokes or CCs belonging to the same symbol, with the edge label matching the symbol class. Spatial relationships are represented by directed labeled with their spatial relationship type. Relation types include *Horizontal*, *Rsub* (right subscript), *Rsup* (right superscript), *Upper*, *Under*, and *PUNC* (for punctuation).

4.2 Multi-task CNN for Classifying Primitives and Primitive Pairs

Image features for primitives and edges in the LOS graph are passed through a CNN architecture to obtain class probability distributions for three tasks: symbol detection (segmentation), symbol classification, and spatial relationship classification. LGAP's CNN architecture is based on QD-GGA [20], an end-to-end trainable multi-task learning model. CNN features are used to estimate probability distributions for each task. These distributions are stored in the cells of three adjacency matrices defined over the input LOS graph. Symbol classification probabilities for primitives are represented along the LOS adjacency matrix

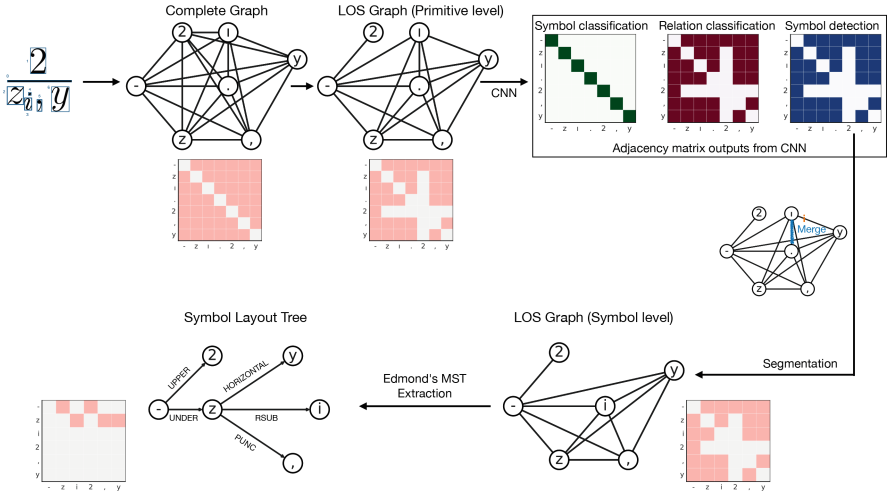


Fig. 4. LGAP formula parsing example. A complete graph over input primitives (here, CCs) is pruned, sub-images corresponding to CCs and CC LOS edges are given symbol, merge/split, and spatial relationship class distributions. Based on merge/split probabilities primitives are merged into symbols (here, for the 'i'), and finally an SLT is produced from remaining spatial relationship edge by extracting a **directed MST** (arrows omitted for legibility). Symbol and relationship class probabilities are averaged when merging primitives into symbols.

diagonal, and merge/split and spatial relationship probabilities are represented for directed LOS edges in off-diagonal entries, as seen at top-right in Fig. 4.

The CNN contains an SE-ResNext backbone [12,38] used to compute a global feature map from the input formula image along with attention modules to produce attention relevance maps from the binary masks described in the previous Section. The SE-ResNext backbone and the attention modules are trained concurrently using a combined cross entropy loss function for all three classification tasks. Attention modules are used to produce 2D relevance maps from the 2D binary masks for nodes, edges and LOS neighbors separately by convolving them through 3 convolutional blocks, trained for each task, where each block has 3 kernels of size 7×7, 5×5, and 5×5 [20].

Modification: For this work, we modified the QD-GGA SE-ResNext encoder by reducing the number of output channels from 512 to 256, to reduce the number of parameters requiring training. This results in a 256-dimensional feature vector for symbol classification, and two 512-dimensional feature vectors for segmentation and relation classification, respectively.

Spatial Pyramidal Pooling and 1-D Context Module. Average pooling uses a single average activation value to represent the convolutional response in a region [1]. Without the use of windowing within an input image, a large amount of important spatial information is lost during average pooling. Jose et

al. [15] use a pyramidal pooling method, which integrates spatial information into the feature vectors, producing more compact and location invariant feature vectors. He et al. [11] introduced the spatial pyramid pooling (SPP) strategy for deep CNNs by adopting the widely used spatial pyramid matching (SPM) strategy [17]. SPP captures spatial information by pooling features within equal-sized regions of the feature map for increasing numbers of sub-regions, forming a pyramid of overlapping sub-regions (e.g., whole image, left-right, top-down, 3 horizontal regions, etc.).

In LGAP and QD-GGA, weighted feature maps are pooled to produce feature vectors. A '1D context' module then performs a 1-by-3 convolution along the sequence of query feature vectors used as classification input [20]. The convolution aggregates features from the previous $(i-1)^{th}$ and next $(i+1)^{th}$ query in input order for the i^{th} query. In the input, queries are spatially sorted top-down, left-right by the top-left coordinate of a query's bounding box. Feature vectors are passed to one of three fully-connected output layers for three classification tasks: segmentation and relationships (edges), and symbol classification (nodes).

Modification: For LGAP, Spatial Pyramidal Pooling is used to capture spatial information across multiple horizontal and vertical regions, providing more spatial information and lower variance in features. We use 5 levels with 11 regions in pooling outputs: this includes 1 full feature map, 2 vertical bins, 3 vertical bins, 2 horizontal bins, and 3 horizontal bins. To reduce the growth in parameters due to increasing pooling regions from 1 to 11, we also reduce the number of output channels in the SE-ResNext encoder from 512 to 64 (a factor of 8). The resulting node and edge feature vectors have lengths of 704 (i.e., 64×11) and 1408 (i.e., 128×11), respectively. With the new LOS attention masks, the feature lengths are 1408 (i.e., 704×2) and 2816 (i.e., 1408×2) respectively. The edge features have an additional factor of 2 due to the concatenation of parent primitive attention masks, as mentioned earlier. We also examine removing the 1-by-3 convolutional context module, since its notion of neighbor is not based directly on spatial proximity as discussed earlier (see Sect. 5).

Multi-task Cross Entropy Loss Function and Training. We use the same cross entropy loss (CE) with a softmax activation used in QD-GGA [20], which combines the segmentation, symbol classification and relation classification losses as shown in Eq. 1. In the equation, δ denotes the total loss, N is the set of primitive nodes (strokes or CCs), E is the set of LOS edges in the input graph, D is the set of segmentation ground truth edge labels, R is the set of relationship ground truth edge labels, and S is the set of primitive ground truth symbol labels.

$$\delta(N,E) = \sum_{e=1}^{|E|}(CE(e,D) + CE(e,R)) + \sum_{n=1}^{|N|} CE(n,S) \tag{1}$$

For backpropagation, LGAP uses an Adam optimizer with learning rate 0.0005 and momentum 0.9, with a weight decay (L2 norm) of 0.1 for regularization.

Table 1. Effect of modifying *PUNC* relationship representation for InftyMCCDB-2. F1 and expression rate metrics are defined in Sect. 3

	Symbol (F1)		Relation (F1)		Expression Rate	
	Detect	+Class	Detect	+Class	Structure	+Class
Original relationships	**98.23**	95.21	94.63	94.28	89.21	81.77
Modified PUNC	98.22	**95.23**	**94.93**	**94.56**	**90.45**	**83.00**

4.3 Parsing: Transforming LOS Graphs into SLTs

After LOS nodes and edges have class distributions assigned to them using the multi-task CNN model, we select all LOS segmentation edges with a higher probability for 'merge' than 'split,' and then merge all connected components over primitives in the resulting segmentation graph into symbols. For the example in Fig. 4, primitives corresponding to the symbol 'i' are merged into a single symbol node, along with their corresponding relationship edges.

For merged primitives, we average the symbol and merged edge relationship class probability distributions. We then apply Edmond's arborescence algorithm [9] to obtain a *directed* Maximum Spanning Tree, which forms the Symbol Layout Tree (SLT). The MST maximizes the sum of relationship probabilities in selected directed edges, where the maximum probability relationship is used for each edge. However, this selection may result in duplicate edges of same relationship type between one parent symbol and two or more child symbols. To address this we apply an additional constraint, where a parent symbol may not have more than one edge of each spatial relationship type associated with it (e.g., to prevent having two horizontal relationships from one symbol to two symbols at right). We replace edges that duplicate a relationship by removing the lower-probability edge, and replacing the lower probability edge's label with its next-highest relationship class probability, and then rebuild the MST, repeating until this unique relationship constraint is satisfied.

5 Experiments

In this Section we report some experiments testing the effects of our modifications to the representation of punctuation relationships in ground truth, and changes in the CNN model and visual feature representation. Experiments were run on two servers and two desktop machines, all with hard drives (HDD):

1. 2 x GTX 1080 Ti GPUs (12 GB), 8-core i7-9700KF (3.6 GHz), 32 GB RAM
2. 2 x GTX 1080 Ti GPUs (12 GB), 12-core i7-8700K (3.7 GHz), 32 GB RAM
3. 4 x RTX 2080 Ti (12 GB), 32-core Xeon E5-2667 v4 (3.2 GHz), 512 GB RAM
4. A40 (48 GB), 64-core Xeon Gold 6326 (2.9 GHz), 256 GB RAM

5.1 Effect of Modifying PUNC Relationship in Ground Truth

We saw a 1.23% improvement in formula recognition rate (Structure + Classification) as seen in Table 1 after applying the modification to ground truth described

Table 2. Effect of LGAP Spatial Pyramidal Pooling (SPP) and 1D context module. Feature vector sizes given as *(node-size, edge-size)*. Modified PUNC representation used

Feature Vectors			Symbol (F1)		Relation (F1)		Expression Rate	
Pool	Sizes	1D-context	Detect	+Class	Detect	+Class	Structure	+Class
Avg	64,128	True	96.99	90.24	91.18	90.62	85.58	66.41
SPP-Avg	704,1408	True	**98.30**	**95.25**	**94.50**	**94.09**	**88.64**	**80.78**
Avg	64,128	False	89.84	34.68	67.68	65.23	64.32	17.57
SPP-Avg	704,1408	False	95.64	87.55	86.16	84.49	77.76	68.80

Table 3. Effect of Adding LOS Neighborhood Masks to LGAP SPP-Avg Model. Original ground truth used (without PUNC modification)

Feature Vectors			Symbol (F1)		Relation (F1)		Expression Rate	
LOS	Sizes	1D-Context	Detect	+Class	Detect	+Class	Structure	+Class
False	704,1408	True	97.96	94.48	94.36	93.89	88.70	78.90
True	1408,2816	True	98.32	**95.66**	**94.85**	94.35	89.27	**83.27**
False	256,512	True	98.23	95.21	94.63	94.28	89.21	81.77
True	512,1024	True	**98.39**	95.49	**94.85**	**94.46**	**89.36**	81.89
False	256,512	False	95.16	83.78	86.48	85.22	79.72	65.26
True	512,1024	False	95.40	85.97	86.27	85.07	80.23	70.09

in Sect. 3. Outputs were produced using a reduced encoder architecture with 256 channels and a context module as described in Sect. 4.2. Observing errors for the two conditions using the `confHist` tool in LgEval showed the new representation correctly identified more punctuation relationships that were previously missing or incorrectly labeled.

5.2 Improving Visual Features: SPP and Increased LOS Context

We next check whether our proposed changes in visual features described in Sect. 4 improve accuracy.

Spatial Pyramidal Pooling. Table 2 shows that when the 1D context module is used, the formula recognition rate improves 14.37% after replacing average pooling by spatial pyramidal pooling (first two rows in Table 2). This difference in expression rate increases dramatically to 51.23% when the context module is removed (last two rows in Table 2). These results suggest that spatial pyramidal pooling greatly improves visual features, allowing us to obtain recognition rates close to the QD-GGA model with 256 encoder channels using only 64 encoder channels. However, removing the 1-D context module reduces the expression rate using the new SPP features by 13.17%, and the original single-region average pooling features reduces dramatically (48.83%). This demonstrates that the new SPP features are beneficial, and the importance of context.

Adding LOS Context through Neighborhood Embeddings. Next we evaluate the impact of including additional LOS neighbors in selecting image regions for queries in the attention modules, as outlined in Sect. 4.1. We hypothesized

Table 4. Benchmarking MST-based Parsing Models on InftyMCCDB-2

MST Model	Symbols		Relationships		Formulas	
	Detect.	+Class	Detect.	+Class	Structure	+Class
LGAP (this paper)	98.32	95.66	94.85	94.35	89.27	83.27
QD-GGA	98.50	94.54	94.43	93.96	87.77	76.72
LPGA_{RF}	99.34	98.51	97.83	97.56	**93.81**	90.06
LPGA_{CNN}	**99.35**	**98.95**	**97.97**	**97.74**	93.37	**90.89**

that incorporating the context from LOS neighbors would reduce ambiguity for visually similar symbols/relationships. Experiments were performed using both the 64-channel encoder output with spatial pyramidal pooling (using 11 bins, resulting a feature vector of lengths 1408 and 2816 (when LOS context is used), as well as the original 256-channel output with average pooling and the original representation. We also investigated the effect of using LOS masks when the context module was removed. The results in Table 3 show improvements in recognition rates at the symbol, relation, and formula levels when LOS neighborhood masks are used. Further, the expression rate accuracy is increased 2.49% over the best SPP model in Table 2.

Error Analysis. An error analysis using the `confHist` tool in LgEval on our best performing model (LOS context, spatial pyramidal pooling, and 1D context; second row in Table 3) reveals the majority of symbol classification errors occur between visually similar symbols, such as $(i, j), (m, n), (l, 1)$, (α, a), and (Left-Paranthesis, Vertical), in decreasing order of frequency. This highlights needed improvements in local visual features for symbols. For relationships, the most frequent errors are missing relationship edges in wide formulas containing a large number of symbols. This type of error can be attributed to the preprocessing step used for inputs: with the height of all formulas fixed at 64 pixels and the aspect ratio preserved, image resolution is noticeably reduced for wide formulas. This leads to features extracted from low-resolution input images being used to locate spatial relationships between connected components for very wide formulas.

We also note that expression rates are influenced by small changes in symbol and relationship recognition accuracy, which may amplify expression rates differences between conditions. For example, if a formula has just one symbol or relationship wrong, it is not counted as correct in the expression rate.

5.3 Benchmarking MST-Based Parsers

As seen in the previous experiments, the LGAP model that obtained the highest expression rate used a combination of Spatial Pyramidal Pooling, line-of-sight attention masks, and a modified punctuation representation in ground truth. We next benchmark this best LGAP model against previous MST-based visual parsers applied to InftyMCCDB-2. Results are presented in Table 4. Performance for LGAP relative to the QD-GGA model it extends is better in every metric, aside from a very small decrease in symbol detection F1 (-0.18%). The

expression rate has increased 6.55% over QD-GGA. Unfortunately, performance is weaker than the LPGA models, with an expression rate that is 7.62% lower.

Despite LGAP's slightly weaker performance than LPGA [19], the LGAP offers substantial benefits in speed and efficiency. The encoder and attention modules are trained end-to-end on a joint loss for multiple tasks in a single feed-forward pass, making the training and execution process much faster than LPGA. Running on a desktop system with two GTX 1080 Ti GPUs (12 GB), an 8-core i7-9700KF processor (3.6 GHz) and 32 GB RAM, LGAP requires 25 min per epoch to train on 12,551 training images and 11 min, 12 s to process the 6,830 test formula images (98.4 ms per formula).

Opportunities for further improvements include improving context usage through graph neural networks, as well as more sophisticated graph-based attention models to replace the current 1D context module from QD-GGA.

6 Conclusion

We have introduced the Line-of-sight with Graph Attention Parser (LGAP) that enhances the visual feature representations employed in the MST-based QD-GGA parser through thoughtful use of Line-of-sight neighborhoods and spatial pyramidal pooling, and modified the ground truth representation of spatial relationship edges connecting punctuation with parent symbols in Symbol Layout Trees (SLTs). These modifications have added contextual information, prevented the loss of spatial information than using single region average pooling in QD-GGA produced, and avoided pruning valid punctuation relationships.

In the future, we aim to improve the use of context. Currently context is introduced using a sequential (1D) module that aggregates the two immediate neighbors in the input order. This sometimes misses neighbor relationships and introduces spurious ones because a spatial sorting order rather than actual proximity defines neighbors. We expect that using Graph Neural Networks (GNNs) will avoid these problems, and incorporate actual proximity in aggregation and use the underlying graph structure directly. Additionally, we plan to replace LGAP's attention maps using more sophisticated methods than simple convolutional blocks. We will also assess our model's performance on online handwritten data, such as CROHME. LGAP is available as open source (see Sect. 4).

Acknowledgements. We thank everyone who contributed to the DPRL formula extraction pipeline: Lei Hu, Michael Condon, Kenny Davila, Leilei Sun, Mahshad Mahdavi, Abhisek Dey, and Matt Langsenkamp. This material is based upon work supported by the Alfred P. Sloan Foundation under Grant No. G-2017-9827 and the National Science Foundation (USA) under Grant Nos. IIS-1717997 (MathSeer project) and 2019897 (MMLI project).

References

1. Akhtar, N., Ragavendran, U.: Interpretation of intelligence in CNN-pooling processes: a methodological survey. Neural Comput. Appl. **32**(3), 879–898 (2019). https://doi.org/10.1007/s00521-019-04296-5
2. Alvaro, F., S'nchez, J., Benedi, J.: Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars. In: 2011 International Conference on Document Analysis and Recognition, pp. 1225–1229, September 2011. https://doi.org/10.1109/ICDAR.2011.247. ISSN: 2379-2140
3. Amador, B., Langsenkamp, M., Dey, A., Shah, A.K., Zanibbi, R.: Searching the ACL anthology with math formulas and text. In: Proceedings ACM SIGIR (2023, to appear)
4. Anderson, R.H.: Syntax-directed recognition of hand-printed two-dimensional mathematics. In: Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc., Symposium, pp. 436–459. Association for Computing Machinery, New York, NY, USA, August 1967. https://doi.org/10.1145/2402536.2402585
5. Baker, J.B., Sexton, A.P., Sorge, V.: A linear grammar approach to mathematical formula recognition from PDF. In: Carette, J., Dixon, L., Coen, C.S., Watt, S.M. (eds.) CICM 2009. LNCS (LNAI), vol. 5625, pp. 201–216. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02614-0_19
6. Chan, K.F., Yeung, D.Y.: Mathematical expression recognition: a survey. Int. J. Doc. Anal. Recogn. **3**(1), 3–15 (2000). https://doi.org/10.1007/PL00013549
7. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational geometry. In: de Berg, M., Cheong, O., van Kreveld, M., Overmars, M. (eds.) Computational Geometry: Algorithms and Applications, pp. 1–17. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77974-2_1
8. Diaz, Y., Nishizawa, G., Mansouri, B., Davila, K., Zanibbi, R.: The MathDeck formula editor: interactive formula entry combining latex, structure editing, and search. In: CHI Extended Abstracts, pp. 192:1–192:5. ACM (2021)
9. Edmonds, J.: Optimum branchings. J. Res. Nat. Bureau Stan. Sect. B Math. Math. Phys. **71B**(4), 233 (1967). https://doi.org/10.6028/jres.071B.032. https://nvlpubs.nist.gov/nistpubs/jres/71B/jresv71Bn4p233_A1b.pdf
10. Eto, Y., Suzuki, M.: Mathematical formula recognition using virtual link network. In: Proceedings of Sixth International Conference on Document Analysis and Recognition, pp. 762–767, September 2001. https://doi.org/10.1109/ICDAR.2001.953891
11. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 346–361. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10578-9_23
12. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7132–7141, June 2018. https://doi.org/10.1109/CVPR.2018.00745

13. Hu, L., Zanibbi, R.: MST-based visual parsing of online handwritten mathematical expressions. In: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 337–342, October 2016. https://doi.org/10.1109/ICFHR.2016.0070. ISSN: 2167-6445
14. Hu, L., Zanibbi, R.: 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), pp. 180–186, October 2016. https://doi.org/10.1109/ICFHR.2016.0044. ISSN: 2167-6445
15. Jose, A., Lopez, R.D., Heisterklaus, I., Wien, M.: Pyramid pooling of convolutional feature maps for image retrieval. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 480–484, October 2018. https://doi.org/10.1109/ICIP.2018.8451361. ISSN: 2381-8549
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations, p. 14 (2017). https://openreview.net/forum?id=SJU4ayYgl
17. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), vol. 2, pp. 2169–2178, June 2006. https://doi.org/10.1109/CVPR.2006.68. ISSN: 1063-6919
18. Li, B., et al.: When counting meets HMER: counting-aware network for handwritten mathematical expression recognition. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. LNCS, pp. 197–214. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19815-1_12
19. Mahdavi, M., Condon, M., Davila, K., Zanibbi, R.: LPGA: line-of-sight parsing with graph-based attention for math formula recognition. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 647–654. IEEE, Sydney, Australia, September 2019. https://doi.org/10.1109/ICDAR.2019.00109. https://ieeexplore.ieee.org/document/8978044/
20. Mahdavi, M., Sun, L., Zanibbi, R.: Visual parsing with query-driven global graph attention (QD-GGA): preliminary results for handwritten math formula recognition. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2429–2438. IEEE, Seattle, WA, USA, June 2020. https://doi.org/10.1109/CVPRW50498.2020.00293. https://ieeexplore.ieee.org/document/9150860/
21. Mahdavi, M., Zanibbi, R., Mouchere, H., Viard-Gaudin, C., Garain, U.: ICDAR 2019 CROHME + TFD: competition on recognition of handwritten mathematical expressions and typeset formula detection. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 1533–1538. IEEE, Sydney, Australia, September 2019. https://doi.org/10.1109/ICDAR.2019.00247. https://ieeexplore.ieee.org/document/8978036/
22. Mansouri, B., Novotný, V., Agarwal, A., Oard, D.W., Zanibbi, R.: Overview of ARQMath-3 (2022): third CLEF lab on answer retrieval for questions on math (working notes version). In: CLEF (Working Notes). CEUR Workshop Proceedings, vol. 3180, pp. 1–27. CEUR-WS.org (2022)
23. Mouchère, H., Zanibbi, R., Garain, U., Viard-Gaudin, C.: Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014. Int. J. Doc. Anal. Recogn. (IJDAR) **19**(2), 173–189 (2016). https://doi.org/10.1007/s10032-016-0263-5

24. Nguyen, C.T., Truong, T.-N., Nguyen, H.T., Nakagawa, M.: Global context for improving recognition of online handwritten mathematical expressions. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12822, pp. 617–631. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86331-9_40
25. Nishizawa, G., Liu, J., Diaz, Y., Dmello, A., Zhong, W., Zanibbi, R.: MathSeer: a math-aware search interface with intuitive formula editing, reuse, and lookup. In: Jose, J.M., et al. (eds.) ECIR 2020. LNCS, vol. 12036, pp. 470–475. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45442-5_60
26. Peng, S., Gao, L., Yuan, K., Tang, Z.: Image to LaTeX with graph neural network for mathematical formula recognition. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12822, pp. 648–663. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86331-9_42
27. Sakshi, Kukreja, V.: A retrospective study on handwritten mathematical symbols and expressions: classification and recognition. Eng. Appl. Artif. Intell. **103**, 104292 (2021). https://doi.org/10.1016/j.engappai.2021.104292
28. Sasarak, C., et al.: min: a multimodal web interface for math search. In: Proceedings Human-Centered Information Retrieval (HCIR), Cambridge, MA, USA (2012). https://www.cs.rit.edu/~rlaz/files/HCIRPoster2012.pdf
29. Shah, A.K., Dey, A., Zanibbi, R.: A math formula extraction and evaluation framework for PDF documents. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12822, pp. 19–34. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86331-9_2
30. Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T.: INFTY: an integrated OCR system for mathematical documents. In: Proceedings of the 2003 ACM symposium on Document engineering, DocEng 2003, pp. 95–104. Association for Computing Machinery, New York, NY, USA, November 2003. https://doi.org/10.1145/958220.958239
31. Suzuki, M., Uchida, S., Nomura, A.: A ground-truthed mathematical character and symbol image database. In: ICDAR, pp. 675–679. IEEE Computer Society (2005)
32. Tang, J.M., Wu, J.W., Yin, F., Huang, L.L.: Offline handwritten mathematical expression recognition via graph reasoning network. In: Wallraven, C., Liu, Q., Nagahara, H. (eds.) Pattern Recognition. LNCS, pp. 17–31. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-02375-0_2
33. Toyota, S., Uchida, S., Suzuki, M.: Structural analysis of mathematical formulae with verification based on formula description grammar. In: Bunke, H., Spitz, A.L. (eds.) DAS 2006. LNCS, vol. 3872, pp. 153–163. Springer, Heidelberg (2006). https://doi.org/10.1007/11669487_14
34. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations, February 2018
35. Wang, J., Du, J., Zhang, J., Wang, B., Ren, B.: Stroke constrained attention network for online handwritten mathematical expression recognition. Pattern Recogn. **119**, 108047 (2021). https://doi.org/10.1016/j.patcog.2021.108047
36. Wu, C., et al.: TDv2: a novel tree-structured decoder for offline mathematical expression recognition. Proc. AAAI Conf. Artif. Intell. **36**(3), 2694–2702 (2022). https://doi.org/10.1609/aaai.v36i3.20172
37. 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. In: Association for the Advancement of Artificial Intelligence, p. 9 (2021). https://www.aaai.org/AAAI21Papers/AAAI-3268.WuJW.pdf

38. Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5987–5995. IEEE, Honolulu, HI, July 2017. https://doi.org/10.1109/CVPR.2017.634
39. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML2015, pp. 2048–2057. JMLR.org, Lille, France, July 2015
40. Yu, D., et al.: Towards accurate scene text recognition with semantic reasoning networks. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12110–12119, June 2020. https://doi.org/10.1109/CVPR42600.2020.01213. ISSN: 2575-7075
41. Zanibbi, R., Blostein, D., Cordy, J.R.: Recognizing mathematical expressions using tree transformation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(11), 1455–1467 (2002). https://doi.org/10.1109/TPAMI.2002.1046157. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence
42. Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., Davila, K.: NTCIR-12 MathIR task overview. In: NTCIR. National Institute of Informatics (NII) (2016)
43. Zanibbi, R., Blostein, D.: Recognition and retrieval of mathematical expressions. Int. J. Doc. Anal. Recogn. (IJDAR) **15**(4), 331–357 (2012). https://doi.org/10.1007/s10032-011-0174-4
44. Zanibbi, R., Mouchère, H., Viard-Gaudin, C.: Evaluating structural pattern recognition for handwritten math via primitive label graphs. In: Document Recognition and Retrieval XX, vol. 8658, p. 865817. International Society for Optics and Photonics, February 2013. https://doi.org/10.1117/12.2008409
45. Zanibbi, R., Yu, L.: Math spotting: retrieving math in technical documents using handwritten query images. In: ICDAR, pp. 446–451. IEEE Computer Society (2011)
46. Zhang, J., Du, J., Dai, L.: Track, attend, and parse (TAP): an end-to-end framework for online handwritten mathematical expression recognition. IEEE Trans. Multimedia **21**(1), 221–233 (2019). https://doi.org/10.1109/TMM.2018.2844689. Conference Name: IEEE Transactions on Multimedia
47. Zhang, J., Du, J., Yang, Y., Song, Y.Z., Dai, L.: SRD: a tree structure based decoder for online handwritten mathematical expression recognition. IEEE Trans. Multimedia **23**, 2471–2480 (2021). https://doi.org/10.1109/TMM.2020.3011316
48. Zhang, X., Gao, L., Yuan, K., Liu, R., Jiang, Z., Tang, Z.: A symbol dominance based formulae recognition approach for PDF documents. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 01, pp. 1144–1149, November 2017. https://doi.org/10.1109/ICDAR.2017.189. ISSN: 2379-2140
49. Zhelezniakov, D., Zaytsev, V., Radyvonenko, O.: Online handwritten mathematical expression recognition and applications: a survey. IEEE Access **9**, 38352–38373 (2021). https://doi.org/10.1109/ACCESS.2021.3063413
50. Zie, Y., Mouchére, H., et al.: ICDAR CROHME 2023: competition on recognition of handwritten mathematical expressions. In: Proceedings ICDAR (2023) (in this proceedings, to appear)