Visual Parsing with Query-Driven Global Graph Attention (QD-GGA): Preliminary Results for Handwritten Math Formula Recognition

Mahshad Mahdavi, Leilei Sun, and Richard Zanibbi
Rochester Institute of Technology
1 Lomb Memorial Dr, Rochester, NY
{mxm7832, lxs5619, rxzvcs}@rit.edu

Abstract

We present a new visual parsing method based on convolutional neural networks for handwritten mathematical formulas. The Query-Driven Global Graph Attention (QD-GGA) parsing model employs multi-task learning, and uses a single feature representation for locating, classifying, and relating symbols. First, a Line-Of-Sight (LOS) graph is computed over the handwritten strokes in a formula. Second, class distributions for LOS nodes and edges are obtained using query-specific feature filters (i.e., attention) in a single feed-forward pass. Finally, a Maximum Spanning Tree (MST) is extracted from the weighted graph. Our preliminary results show that this is a promising new approach for visual parsing of handwritten formulas. Our data and source code are publicly available.

1. Introduction

Mathematical notation is an essential source of information in many fields and the ability to recognize them is an important module in OCR (Optical Character Recognition). Math recognition is vital for several research-based and commercial-based applications, such as educational aids, navigational tools, digital assistance, and other tasks that require machines to understand mathematical notation [1].

Recognizing math requires inferring a formula representation from the input (e.g. raster images, strokes, or PDFs) that identifies symbols and their relationships. Formulas are represented using trees: in fact, MathML and LaTeX are trees with additional formatting annotations. Formulas are generally represented visually by Symbol Layout Trees (SLTs) giving symbols and their placement on writing lines, or semantically by Operator Tree (OPTs) describing mathematical content (i.e., quantities and operations [2]).

Our work belongs to the family of visual parsers for math that label and prune a graph over input primitives (e.g., handwritten strokes [3], or connected components in images [4, 5]). Graph-based parsing for math does not require an expression grammar - the language model consists of only node and edge labels applied to directed, rooted trees. Defining the parser input with graphs frees us from forcing the input into a sequence over the 2D input image, as commonly done for state-of-the-art RNN-based models [6, 7].

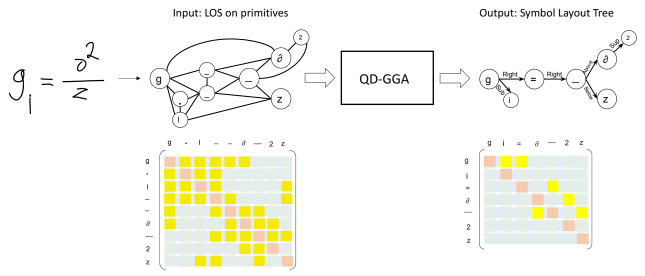

Figure 1. Input and output graphs for $g_i = \frac{\delta^2}{z}$. Adjacency matrices for stroke-level input and symbol-level output are shown.

Our novel Query-Driven Global Graph Attention (QD-GGA) parser extends traditional CNN models designed for sequential and multi-dimensional data to graph-structured data (see Figure 1). QD-GGA is comprised of the four modules shown in Figure 2: attention, feature extraction, classification, and MST extraction. All class distributions for nodes and edges in a Line-of-Sight (LOS) graph are predicted at each feed-forward pass, with the help of attention-based filtering of features defined using the LOS graph. A Symbol Layout Tree (SLT) is extracted from the resulting weighted graph, as a maximal spanning tree.

In the remainder of the paper, we introduce related work (Section 2), define the QD-GGA model (Section 3), present results on the CROHME data set (Section 4), and discuss future work (Section 5).

Contributions:

1. End-to-end structure learning directly from a joint loss

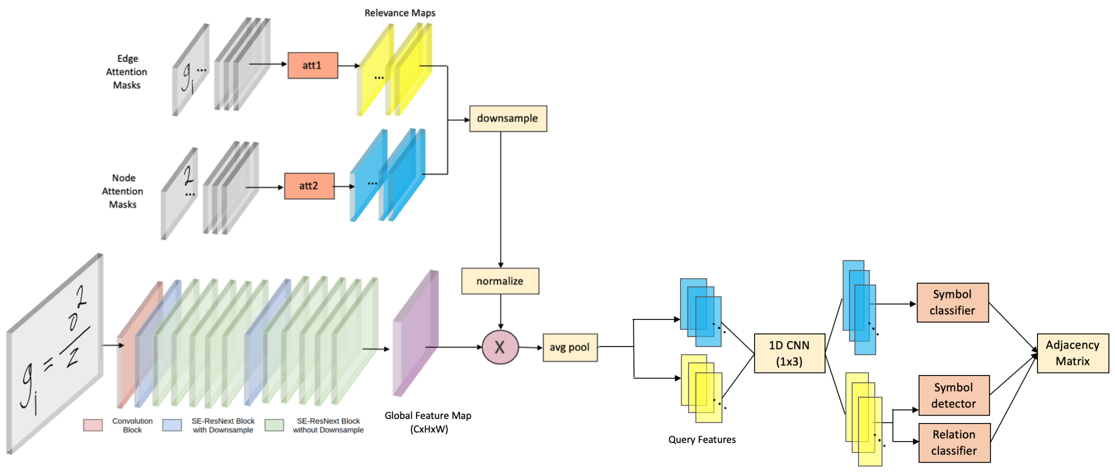

Figure 2. QD-GGA Architecture. The dimensions of the 2D features are channel, height and width. The dimensions of the linear features are batch size and channel. The batch size is N which is the number of binary masks or queries (all nodes and edges) to answer (shown in gray). The final adjacency matrix has class distributions for symbols on the diagonal, and two distributions for segmentation and relationship labels for each edge.

computed over adjacency matrices holding class distributions for stroke and stroke-pair class labels.

2. A novel attention model that *queries* a shared feature representation to efficiently obtain inputs for multiple classification, segmentation, and relationship decisions, vs. computing features/attention dynamically over an image (e.g., in RNN models [6, 7]).
3. QD-GGA generalizes our previous work using CNN-based features [4], with features and attention modules trained concurrently for multiple tasks.

2. Related Work

In the following we provide an overview of approaches proposed for structure parsing, with a focus on methods pertinent to formula recognition.

2.1. Visual Parsing

Visual parsing is important for a variety of image understanding tasks, including real-world complex vision-language tasks such as image caption generation [8, 9], visual relationship detection [10, 11, 12], scene graph generation [13, 14, 15], and table detection and form parsing [16, 17]. We focus only on graph parsing approaches for general structure learning in this section.

Dai et al. [12] use a graph parsing method to detect visual relationships by first doing an object detection to generate the nodes from the input image. Second, producing a set of object pairs from the detected objects to define relationships between objects. With n detected objects, they form a complete graph of $n(n-1)$ pairs (edges), with some unlikely relations filtered out with a low-cost neural net. Each retained pair of objects will be fed to the joint recognition module. Taking into account multiple factors and their relations, this module will produce a triplet in the form of (subject, predicate, object), as the output.

Inspired by this work, [18] use Graph Convolutional Networks encoder plus Long Short-Term Memory decoder (dubbed as GCN-LSTM) architecture to encode both semantic and spatial object relationships. In this work, salient image regions (nodes) are computed implementing a Faster R-CNN [19]. Second, two directed graphs are generated on the detected regions, one for spatial relations (e.g. inside, overlap, over, etc.) and another one for semantic relations (e.g. riding, eating, biking, etc.). Graph Convolutional Networks (GCN) are then exploited to encode region representations and visual relationship in both graphs. Next, the learnt relation-aware region representations are feed into individual attention LSTM decoders to generate the sentence. To integrate outputs of two decoders, the predicted score distributions on words from two decoders is averaged at each time step.

The results from the two decoder are fused in an inference stage adopting a late fusion scheme to linearly fuse the results from two decoders. The semantic graph decides which relations should be established between objects, leaving the spatial relations between image regions unused. A second graph defining spatial regions is then generated over the detected regions. When doing classification in the semantic graph, similar to our work, a 'NoRelation' class is added to the set of class labels N_{sem}. They compute the probability distribution on all the $(N_{sem}+1)$ relation classes for each object pair. If the probability of NoRelation is less than 0.5, a directed edge connects the region vertex of parent (subject noun) to the region vertex of child (object noun). The relation class with maximum probability is regarded as the label of this edge.

Yang et al. [13] also use a graph parsing approach for scene understanding. Their proposed model called Graph R-CNN has three modules: (1) object node detection which is done with R-CNN similar to [18], (2) relationship edge pruning with a Relation Proposal Network (RePN), and (3) graph context integration. RePN learns to compute 'relatedness' scores between object pairs which are used to prune unlikely scene graph connections. Next, an attentional graph convolution network (aGCN) is applied on pruned graph to encode higher-order context throughout the graph and provide information on each object and relationship representation based on its neighbors.

Our method differs in three aspects: (1) We use connected components from rendered handwritten strokes as input, generating a Line-of-Sight graph - here nodes are objects to classify and edges are relations to parse. (2) We train our model to do object segmentation, classification and relation prediction jointly using a multi-task classification framework. (3) Unlike [13], our model handles multiple queries (i.e., classification problems) using a soft attention module, so that attention masks generated for each node and node pair are refined through back propagation when training the model end-to-end.

2.2. Math Recognition

Visual parsing of mathematical expressions converts input images to a representation of formula structure which is a hierarchical arrangement of symbols on writing lines. A common set of features used to represent the spatial relations between components (e.g. symbols) are geometric features. Visual features have also been used [20, 21]. In the following, we review the main approaches in math expression recognition.

Syntactic Methods (Constituency Parsing). These methods are useful for interpreting complex patterns in mathematical formulas because the notation has an obvious division into primitives, a recursive structure, and a well-defined syntax [22, 23, 24, 25]. Alvaro et al. [26, 27] present an online handwritten math expression recognition system using Stochastic Context-Free Grammars (SCFG) defined at the symbol level. Segmentation is done by scoring symbol candidates using a symbol classifier and letting the SCFG parser determine whether they should be merged based on confidence scores from an SVM classifier.

Probabilities associated with grammar rules are learned from training data. For parsing, first, lexical units are built from the set of symbol segmentation hypotheses. Second, a set of syntactic and spatial constraints defined by the grammar guide parsing to to identify candidate parse trees, and the most probable expression is returned.

Minimum Spanning Trees (Dependency Parsing). Mathematical expression recognition can be posed as searching for a tree representing symbols and their associated spatial relationships within a graph of primitives. Suzuki et al. [5] present MST-Based math expression recognition system using virtual link networks. Recognition is done by finding a spanning tree for the network with minimum weight.

Another MST-based math parsing method is presented by Hu et al. [3, 28] using Edmond's algorithm to extract a tree from a weighted LOS graph. They use an LOS graph for representing the layout of primitives and symbols in math expressions [29]. LOS graphs have a higher maximum formula tree expressivity than many other graph representations (e.g., Delaunay, geometric MST, k-NN for $k \in \{1 \ldots 6\}$), while also reducing the search space for parsing. They also modify the shape context feature with Parzen window density estimation. These Parzen shape contexts are used for symbol segmentation, symbol classification and symbol layout analysis.

An generalization of the work done by Hu et al. [3] is the LPGA (Line-Of-Sight Parsing with Graph-based Attention) model [4]. In LPGA individual CNNs are trained for segmentation, classification, and parsing. A hierarchical approach is used to first segment nodes into symbols with a binary classifier, and then generate a second graph on symbols and train two separate models to learn symbol classes and spatial relationships.

Encoder-Decoder Models. IM2TEX, inspired by the sequence-to-sequence model designed for image caption generation by Xu et al.[9] directly feeds a typeset formula image generated using $\LaTeX$ into a Convolutional Network to extract a feature grid learned by the network [7]. For each row in the feature map, an RNN is used to encode spatial layout information. The encoded fine features are then passed to an attention-based recurrent neural network decoder, which then emits the final expression string.

Another encoder-decoder model by Zhang et al. [6] use pen traces collected from handwritten strokes on a tablet for parsing. In their model, the encoder is a stack of bidirectional GRUs while the parser (decoder) combines a GRU-based language model and a hybrid attention mechanism consists of a coverage-based spatial attention and a temporal attention. Unlike the attention module in IM2TEX model, which scans the entire input feature map at pixel level, the spatial attention in TAP learns an alignment between input strokes and outputs. The role of temporal attention in TAP model is to learn when to rely on the product of spatial attention and when to just rely on the language model as there is no spatial mapping for tokens representing spatial relationships e.g., superscript '∧' or subscript '_'.

The latest architecture in encoder-decoder networks used for math expression recognition uses two input branches to encode both online and offline features [30]. This Multi-modal Attention Network (MAN), first takes dynamic trajectories and static images for online and offline channels of the encoder respectively. The output of the encoder is then transferred to the multi-modal decoder to generate a LaTeX sequence as the mathematical expression recognition result. This architecture is a hybrid design of both TAP and IM2TEX models explained earlier, except the fact that they use CNN layers in their online channel instead of using a stack of RNNs. Each decoder has their own attention modules. For the online branch, the attention module highlights strokes, whereas in the offline module it weights the pixels. Once the attention weights are calculated, the multi-modal context vector can be obtained by concatenating the single online and offline context vectors. A recent addition to sequential models [31, 32] exploits DenseNet [33] for encoding images. In this work an improved attention model with channel-wise attention is applied before spatial attention.

Motivations for our Approach. Compared to BLSTM in sequential models, graphs are more natural and general for representing math equations as trees. In our model, handwritten strokes form the nodes of the input and output graph, while edges between strokes in the output represent symbol segmentation and spatial relationships (e.g., right, superscript) between symbols. Our attention module uses this input graph to query (filter) CNN features from a single feature representation of the input image to efficiently obtain inputs for multiple classification, segmentation, and relationship decisions.

Our system does not need to learn an alignment between input strokes and outputs similar to encoder-decoder models, as we directly output a graph-based hierarchical representation of symbols on writing lines (as a *Symbol Layout Tree (SLT)*), and not a string. We also do not use expression grammars for language models, instead relying only upon the sets of symbol and relationship classes along with visual statistics captured by our CNN models. The language model in our system consists of only symbol and relationship labels.

3. QD-GGA

Our CNN-based model predicts all node and edge classes at each feed-forward pass with the help of a query-driven attention mechanism. An input LOS graph is computed from handwritten strokes. Then, class probabilities generated by the CNN classifiers are assigned to node and edges in the input graph, symbol segmentation and classification decisions are made, and then Edmond's arborescence algorithm [34] is used to extract a directed tree from the weighted graph over symbols (see Figure 3).

During training, a joint classification loss is calculated from the output matrix, which back-propagates through all three classifiers responsible for segmentation, classification and relationships, as well as the attention layers, enabling us to train them simultaneously with shared features.

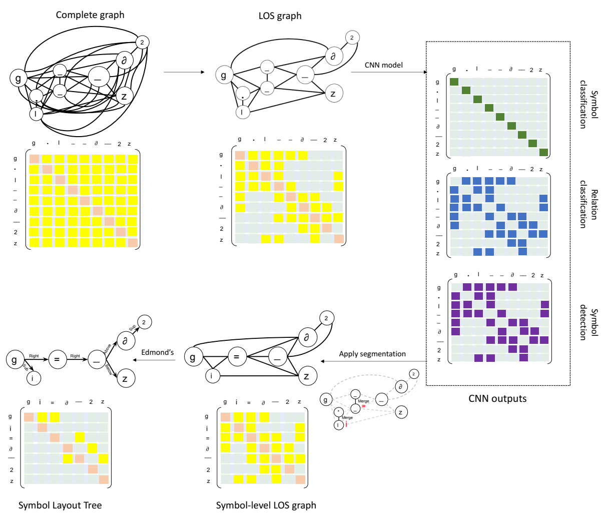

Figure 3. Parsing the formula in Figure 1 ($g_i = \frac{\delta^2}{z}$). Graph edges are yellow adjacency matrix entries; strokes appear in red along diagonals. A stroke-level line-of-sight graph is constructed, after which strokes are classified as symbols, and edges between strokes classified for 1) symbol detection (merge/split), and 2) relation classification. Symbol detection decisions are applied to convert the graph into a symbol level graph, followed by averaging symbol (node) and relationship (edge) scores to obtain a new weighted graph. Finally, Edmond's arborescence algorithm extracts a maximal symbol layout tree.

Our CNN model is shown in Figure 2. Our architecture is modular, independent of the CNN feature model (any feature model such as VGG or ResNet might be used), easy to implement, and faster than recurrent approaches in training.

3.1. Graph Representation

For dependency parsing such as done in QD-GGA, we need to identify a sub-graph with minimal cost or maximum probability. For math recognition, the final subgraph is usually a Symbol Layout Tree. Ideally, we would like to reduce the size of the search space by using a graph that only has edges between the primitives having a spatial relationship in the original setting (perfect precision) and avoid the miscellaneous edges which add confusion to the problem.

A study by Hu et al. [28] on parsing handwritten formulas (see Section 2.2), proposes using Line of Sight (LOS) graphs [35], which represents whether nodes can 'see' one another. LOS graphs can represent roughly 98% of CROHME 2014 dataset formulas, while reducing the number of edges to $3.3n$ for n strokes, many fewer than the $n(n-1)/2$ edges in a complete graph.

In this work, we also also use the Line of Sight (LOS) graphs over handwritten strokes. A stroke represents a drawn line on a writing surface. To work with images, an extra step of rendering strokes is needed to provide the

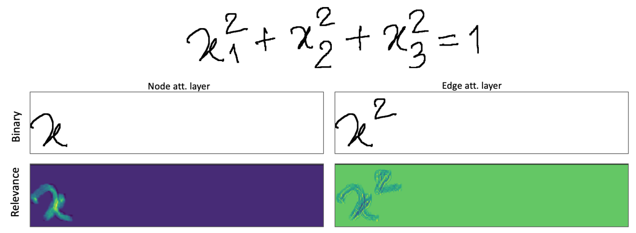

Figure 4. Attention Masks. Input binary masks for node x and edge $(x, 2)$ are shown in the first row. Corresponding real-valued relevance masks applied to features are shown in the second row.

nodes in our graph. Figure 3 shows edges in a complete graph for $g_i = \frac{\delta^2}{z}$ alongside the edges in a LOS graph. Nodes sharing an edge are marked with yellow in the adjacency matrix.

3.2. Network Architecture

The QD-GGA architecture is shown in Figure 2. The network contains a convolution block (shown in red) followed by two SE-ResNext block groups. SE-ResNext blocks combine Squeeze-And-Excitation [36] with a ResNext block [37]. To compute image features, each SE-ResNext block group contains six SE-ResNext blocks with the first SE-ResNext block having a down sample layer (shown in blue). The receptive field of our final feature map is 35 pixels; each point in the final feature map corresponds to a 35×35 region in the input image, with the minimum symbol height set at 64 pixels.

Given a sequence of primitive feature vectors, the vectors are concatenated along the length dimension of a 1D feature tensor. Then, the temporal context module performs a 1-by-3 convolution along the length dimension treating each primitive as an individual element. The convolution operation consolidates features of a primitive neighborhood by considering the $i-1$ and $i+1$ primitives for the i primitive (in time order).

3.3. Attention Module

As seen in Figure 2, there is a side branch consists of two attention layers which takes binary masks for nodes and edges separately as inputs and apply convolution on them. We performed extensive experimental analysis to understand the performance trade-offs amongst different combinations of shared and task-specific representations in the main and side stream. The best configuration has a 3 convolutional blocks with each block having 4 kernels of size of 7×7, 5×5, and 5×5. The final relevance maps are 2D $(H \times W)$ similar to input binary masks, see Figure 4.

The attention module queries a shared CNN feature map to efficiently obtain inputs for all classification, segmentation, and relationship decisions. Spatial masks provide attention, comprised of either individual node binary masks,

Algorithm 1 SLT Extraction from Adjacency Matrix

```
1. Use 'Merge' edges to group primitives into symbols
2. Classify symbols by max. mean score for member strokes
3. Score symbol relationships by avg. stroke pair distributions
4. Apply Edmond's algorithm to obtain maximal SLT
```

or masks from pairs of nodes sharing an edge in the input graph. The attention module takes binary masks and then generates relevance maps (i.e., continuous masks) by convolving binary masks with kernels trained for each task (per [38]). Figure 4 shows binary masks and their corresponding relevance masks for node (single stroke) x and stroke pair $(x, 2)$. Attention relevance maps are downsized and then multiplied with the global feature map. In this way, the downsampled and normalized relevance mask weights the feature map to focus on ('query') the relevant input region. Finally, the weighted feature map is average pooled to a one dimensional feature vector which can be regarded as the extracted feature for the primitive (stroke) or primitive pair (stroke pair).

We normalize attention mask values using the Instance Norm [39] with $\epsilon = 0.001$. This converts values into a measure of standard deviations from the mean $(E[x])$:

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}$$

3.4. SLT Extraction

We use Edmond's algorithm [34] to extract a Maximum Spanning Tree (MST) from class distributions associated with the LOS adjacency matrix output. Experiments demonstrate that it is more accurate to apply symbol segmentation results before extracting relationships (see Figure 5), rather than extract an MST directly from the stroke-level matrix. Algorithm 1 provides the steps used to convert a stroke-level graph to a symbol-level graph. First, primitives that belong to a single symbol are merged based on the segmenter predictions. Symbol class distributions are computed by averaging symbol classifier probabilities over all primitives belonging to a single symbol. Then, all incoming and outgoing edges attached to primitives grouped into a symbol are merged into one incoming and one outgoing edge. Again, probabilities over merged edges are averaged to generate the symbol-level edge probability distributions.

SLT generation is illustrated in Figure 5. In the example, primitives belong to i and $=$ are merged into symbols. All edges connected to these four primitives, shown with blue patches, should be updated in the symbol-level graph. Average probabilities for stroke-level edges provide the distributions for symbol-level merged edges.

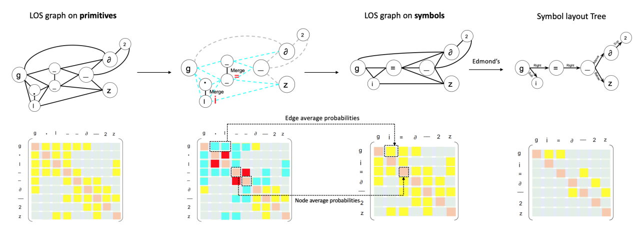

Figure 5. SLT Extraction. A stroke-level graph is converted to symbol-level after applying segmentation results, and then an SLT is extracted using Edmond's arborescence algorithm. Red patches in the middle matrix show primitives to be merged into symbols. Blue patches show edges that should be updated after merging nodes. Merged probabilities for symbols and symbol relationships are averaged over constituent stroke-level elements.

3.5. Implementation and Training

Loss. The loss designed for an Multi-Task Learning (MTL) model should allow tasks contribute to training equally, without letting the easier tasks dominate the learning. We use the cross entropy loss (CE) with a softmax layer to normalize the network outputs. The final loss is the sum of all the errors in segmenter, parser and symbol classifier. The loss function δ is defined in equation 1. Here N is the stroke set (nodes), and E is the set of line-of-sight edges in the adjacency matrix. D is the set of detection ground truth labels for edges, R is the set of relationship ground truth labels for edges, and S is the set of ground truth symbol labels for strokes.

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

Tree Loss. To reduce the effect of edges not contributing to the final tree we introduce "tree loss," where only ground truth edges and false positive edges in the final SLT are counted in loss calculations. Therefore, only hard negatives mistaken for real relations in the output are include in the loss computation. We tried other loss designs, e.g., harmonic mean of individual loss types, weighted combinations of individual losses, loss defined on MST edges, but these experiments showed the tree loss and direct sum (Eq. 1) work best. The models used for experiments in Section 4 are trained with tree loss.

Training Process. Since math expressions have different sizes, we replace the conventional batch normalization layers with group normalization [40] for all the blocks, as this is more robust for small batch sizes. Group Normalization divides the channels (in feature maps) into groups and computes the mean and variance within each group for normalization making the computation independent of batch sizes, and its accuracy stable for a wide range of batch sizes.

The QD-GGA CNN has 13,854,478 parameters. We use an Adam optimizer to learn the parameters. The batch size was set to 1, momentum to 0.9 and the learning rate was initially set to 10^{-2}, and then decreased by a factor of 10 when the validation set accuracy stopped improving. The training was regularized by weight decay set to 0.004. The system is built using PyTorch and experiments were run on an 8GB Nvidia 1080 GPU. Experiments were run on a server with an 8-core Intel Xeon E5-2667 processor (3.20 GHz per core), and 256 GB RAM was available. The time complexity of our model is $O(|N| + |E|)$ for an input graph with E edges and N nodes. In the worst case (complete graph) $|E| = |N| \times (|N| - 1)$.

4. Experiments

In this section, we present results on the CROHME handwritten math dataset, using rendered formula images from the competition. Results are compiled using the LgEval library [41]. We report recognition rates for formulas, and F-scores for detection and classification of symbols and relationships. Results for correct symbol/relationship locations (*Detection*), correct symbol/relationship detections and classes (*Det.+Class*), unlabeled SLT structure, and SLT structure with correct labels (*Str.+Class*) are presented in Tables 1-4. Please note that in each of these tables, symbol and relationship detection results are reported across all formulas in CROHME, while formula recognition rates are reported for complete formulas (i.e., input files). For better comparisons, the similar experiment across Tables 1-3 is indicated with an asterisk (*).

CROHME Dataset. CROHME datasets contain handwritten formulas created online [41, 42]. Stroke data is given as lists of (x,y) coordinates representing points sampled as strokes are written. An InkML format represents strokes and formula structure in Presentation MathML. We render images from the (x,y) points in the CROHME InkML files with the minimum symbol height to be set at 64 pixels and set the width to be the rescaled math expression width. The training, validation and test sets in CROHME 2019 contain 9993, 986, and 1199 expressions, respectively. The dataset includes 101 symbol classes. CROHME 2016 and CROHME 2014 test sets are used for benchmarking in Table 5 each having 1147 and 986 expressions.

4.1. Graph Representations

We first studied the effect of using different input graph representations. In previous studies on handwritten math recognition [28], the Line-Of-Sight graph has shown promising results. Results using LOS graphs for typeset formula images [4] support this observation. However, error analysis in both models shows that a long superscript may block the field of view of two symbols with a connection and cause a missing ground-truth edge in LOS graphs.

To address this, we tried using complete graphs. Table 1 shows our results. We believe LOS graphs are better rep-

Table 1. Effect of graph representations.

	Symbols		Relationships		Formulas	
	Detection	Det.+Class	Detection	Det.+Class	Structure	Str.+Class
*LOS graph	97.81	87.35	89.55	88.16	**58.84**	**31.35**
Complete graph	97.89	87.27	84.41	83.21	46.10	24.81

Table 2. Effect of Attention Models on Recognition Accuracy.

	Symbols		Relationships		Formulas	
	Detection	Det.+Class	Detection	Det.+Class	Structure	Str.+Class
*Binary Mask	97.81	87.35	89.55	88.16	58.84	31.35
Trained Mask (1 kernel)	97.63	88.44	90.13	88.63	60.52	34.62
Trained Mask (3 blocks)	97.43	88.72	91.16	89.83	**62.53**	**36.13**

Table 3. Effect of Parent Stroke Features and Number of Classifiers. Last row: separate classifier for segmentation added (merge/split).

	Symbols		Relationships		Formulas	
	Detection	Det.+Class	Detection	Det.+Class	Structure	Str.+Class
2 class.	73.38	66.18	51.25	50.13	24.06	13.5
2 class. + p.	82.21	74.54	66.99	66.19	40.15	23.30
*3 class. + p.	97.81	87.35	89.55	88.16	**58.84**	**31.35**

resentations since the complete graph contains more edges, resulting in more variation in features and a larger search space for spatial relationships. Our results are computed on the CROHME 2019 test set using a CNN with separate classifiers for symbol classification, stroke segmentation, and stroke relationships.

4.2. Attention Module

We then examined different attention models, trying both static binary masks (hard attention) and trainable attention masks (soft attention) (see Table 2). As we expected, the trainable masks allow attention to be more powerful. It is interesting that with only training one kernel (7×7) per binary mask, formula recognition rates are boosted 3%, and symbol classification and relationship detection both are improved. Additional experiments showed that attention layers with three convolution blocks provides the best performance, and that stacking more blocks does not improve the performance.

4.3. Classification: Features and Tasks

In an earlier design we had two classifiers: one responsible for classifying strokes (nodes), and the other for classifying edges, with "Merge" added to the spatial relationship classes (including "NoRelation"). In this setup, we did not differentiate features generated for directed edges between nodes (i.e., visual features for (A, B) matched those for (B, A)).

Since features were identical for both edge directions, we decided to add features to signal which node is the parent for each edge, by concatenating the stroke pair masked features with the parent stroke masked features when classifying edges. Figure 6 shows that each feature vector for edges is concatenated with the features generated by parent mask. The feature vector for edge classification then

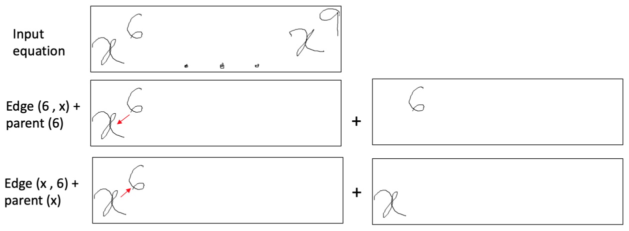

Figure 6. Concatenating parent features with edge pairs to introduce visual difference for directed edges shown with their corresponding binary masks.

Table 4. Comparison against State-of-the-art math recognition models. Expression rates are reported for comparisons.

	CROHME 2014			CROHME 2016					
System	ExpRate	≤ 1	≤ 2	ExpRate	≤ 1	≤ 2	Spatial att.	Temporal att.	Grammar
IM2TEX	35.90	-	-	-	-	-	Yes	Yes	No
TAP	48.47	63.28	67.34	44.81	59.72	62.77	Yes	Yes	Yes
WAP	48.38	66.13	70.18	46.82	64.64	65.48	Yes	Yes	Yes
MAN	54.05	68.76	72.21	50.56	64.78	67.13	Yes	Yes	Yes
LPGA_{RF}	26.88	36.63	42.50	-	-	-	Yes	No	No
QD-GGA	32.04	45.06	55.23	32.84	46.88	59.12	Yes	No	No

becomes 1024 elements, while remaining 512 elements for the node classifier. This makes the parsing more accurate as seen in the first two rows of Table 3.

We then looked at separating the segmentation and relationship decisions into separate tasks, adding a third classifier responsible for symbol detection (segmentation). The merge/split classification is binary, making this an easier task than when adding the 'merge' label to the set of spatial relations and classifying them altogether. Edge features are fed to a segmenter for a binary classification and to a parser for relation classification. This improves both symbol and relationship recognition substantially, as seen in the last row of Table 3.

Finally, to study the benefit of jointly training the classifiers, we compare our results with our baseline model in which individual classifiers are trained for each task in isolation using the same input graph. Experiments on the CROHME 2014 test set shows that jointly training classifiers with our attention module increases the expression rate by 5%, with improvements in both segmentation and spatial relationship classification rates.

Table 5. Benchmarking QD-GGA against CROHME2019 participating systems. Models evaluated using SymLG metrics.

	Structure + Symbol Labels			Structure
CROHME 2019	ExpRate	≤ 1	≤ 2	Correct
USTC-iFLYTEK	**80.73**	**88.99**	**90.74**	**91.49**
Samsung R&D	79.82	87.82	89.15	89.32
MyScript	79.15	86.82	89.82	90.66
QD-GGA	40.65	60.01	64.96	60.22

4.4. Benchmark

We benchmark our proposed model against state-of-the-art systems. Table 4 shows the results of current configu-

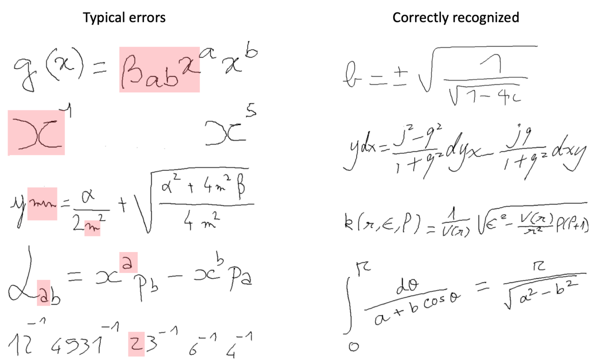

Figure 7. Example QD-GGA Results for CROHME 2019. Column at left shows the typical error cases; large subscripts mistaken with baseline (first row), visually similar symbols classified incorrectly (d instead of a and z instead of 2).

ration of our model against state-of-the-arts on CROHME 2014 and 2016 datasets. Table 5 compares our preliminary results on CROHME 2019 against the winners of the competition. Please notes that recognition rates are not comparable across these two table as the latter computed using SymLG [43]. The main experimental results show that Line-Of-Sight Parsing with Query-Driven Global Graph Attention (QD-GGA) is effective for math expressions recognition.

Figure 6 presents recognition results from QD-GGA, including correctly recognized equations along with examples of common errors. Most structure recognition failures are caused by missing edges in LOS graphs, or incorrect baseline detection as a result of size variations in handwritten characters, e.g., the “subscript” relation between a and β is classified as “right.” Most symbol classification errors are among visually similar classes such as (X,x), (m,n), (α, a), (d,a), (z,2), etc. Lastly, the second row shows a segmentation error in which two strokes in an X have not been merged.

The current model is equipped with a only simple soft attention module which we plan to improve. Although the main results show that we do less well than other systems, we think that there is promise for the approach taken in QD-GGA. Our current model has no temporal attention, or any record of sequential information. We also do not apply any grammatical rules.

The training time reported for Tap [6] system is 780 sec/epoch for the base model and it is even longer to train the ensemble models, whereas the training for our best configuration takes 254 sec/epoch. Execution time reported on CROHME2014 for TAP model is 377 sec, the WAP system [44] and the ensemble of TAP+WAP each takes 196 and 564 sec respectively. The execution time for QD-GGA on the same dataset is 59 sec which is much faster.

5. Conclusion

In this work, we introduced the Query-Driven Global Graph Attention (QD-GGA) parsing model, a new CNN-based variant of graph-based visual parsing. Our novel graph-based attention module allows multiple classification queries for nodes and edges to be computed in one feed-forward pass. By using a Multi Task Learning (MTL) framework, it is possible to train our CNN for different tasks simultaneously from an output adjacency matrix containing class distributions at each entry. This provides generalization for feature representations, and a more global view for classifiers through a shared joint loss.

In the future, we would like to improve our attention modules, and explore adding a temporal attention similar to GCNs to learn from a sequential order over nodes and edges in the input graph. We also would like to apply QD-GGA to similar visual parsing problems e.g., parsing chemical diagrams.

References

[1] R. Zanibbi and D. Blostein, “Recognition and retrieval of mathematical expressions,” *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 4, pp. 331–357, 2012. 1

[2] R. Zanibbi and A. Orakwue, “Math search for the masses: Multimodal search interfaces and appearance-based retrieval,” in *Conferences on Intelligent Computer Mathematics*. Springer, 2015, pp. 18–36. 1

[3] L. Hu and 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. 1, 3

[4] M. Mahdavi, M. R. Condon, and K. Davila, “LPGA: Line-of-sight parsing with graph-based attention for math formula recognition,” 2019. 1, 2, 3, 6

[5] Y. Eto and M. Suzuki, “Mathematical formula recognition using virtual link network,” in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*. IEEE, 2001, pp. 762–767. 1, 3

[6] J. Zhang, J. Du, and L. Dai, “Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition,” *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 221–233, 2018. 1, 2, 3, 8

[7] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, “Image-to-markup generation with coarse-to-fine attention,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 980–989. 1, 2, 3

[8] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3128–3137. 2

[9] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057. 2, 3

[10] L. Peng, Y. Yang, Z. Wang, X. Wu, and Z. Huang, "Cra-net: Composed relation attention network for visual question answering," in *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2019, pp. 1202–1210. 2

[11] C. Han, F. Shen, L. Liu, Y. Yang, and H. T. Shen, "Visual spatial attention network for relationship detection," in *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 2018, pp. 510–518. 2

[12] B. Dai, Y. Zhang, and D. Lin, "Detecting visual relationships with deep relational networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3076–3086. 2

[13] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 670–685. 2, 3

[14] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5831–5840. 2

[15] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5410–5419. 2

[16] B. Davis, B. Morse, S. Cohen, B. Price, and C. Tensmeyer, "Deep visual template-free form parsing," *arXiv preprint arXiv:1909.02576*, 2019. 2

[17] J. Hirayama, H. Shinjo, T. Takahashi, and T. Nagasaki, "Development of template-free form recognition system," in *2011 International Conference on Document Analysis and Recognition*. IEEE, 2011, pp. 237–241. 2

[18] T. Yao, Y. Pan, Y. Li, and T. Mei, "Exploring visual relationship for image captioning," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 684–699. 2, 3

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99. 2

[20] F. Alvaro and R. Zanibbi, "A shape-based layout descriptor for classifying spatial relationships in handwritten math," in *Proceedings of the 2013 ACM symposium on Document engineering*. ACM, 2013, pp. 123–126. 3

[21] R. H. Anderson, "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*. ACM, 1967, pp. 436–459. 3

[22] A. Belaid and J.-P. Haton, "A syntactic approach for handwritten mathematical formula recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 105–111, 1984. 3

[23] S. MacLean and G. Labahn, "A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 2, pp. 139–163, 2013. 3

[24] A.-M. Awal, H. Mouchère, and C. Viard-Gaudin, "A global learning approach for an online handwritten mathematical expression recognition system," *Pattern Recognition Letters*, vol. 35, pp. 68–77, 2014. 3

[25] D. Blostein and A. Grbavec, "Recognition of mathematical notation," in *Handbook of character recognition and document image analysis*. World Scientific, 1997, pp. 557–582. 3

[26] F. Alvaro, J.-A. Sánchez, and J.-M. Benedí, "An integrated grammar-based approach for mathematical expression recognition," *Pattern Recognition*, vol. 51, pp. 135–147, 2016. 3

[27] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models," *Pattern Recognition Letters*, vol. 35, pp. 58–67, 2014. 3

[28] L. Hu and 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. 3, 4, 6

[29] "Details omitted for anonymous review - not a peer-reviewed publication," 2017. 3

[30] J. Wang, J. Du, J. Zhang, and Z.-R. Wang, "Multi-modal attention network for handwritten mathematical expression recognition." 3

[31] J. Wang, Y. Sun, and S. Wang, "Image to latex with densenet encoder and joint attention," *Procedia computer science*, vol. 147, pp. 374–380, 2019. 4

[32] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Handwritten mathematical expression recog-

nition via paired adversarial learning," *International Journal of Computer Vision*, pp. 1–16, 2020. 4

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708. 4

[34] J. Edmonds, "Optimum branchings," *Journal of Research of the national Bureau of Standards B*, vol. 71, no. 4, pp. 233–240, 1967. 4, 5

[35] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, "Computational geometry: introduction," *Computational Geometry: Algorithms and Applications*, pp. 1–17, 2008. 4

[36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141. 5

[37] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500. 5

[38] S. Eppel, "Setting an attention region for convolutional neural networks using region selective features, for recognition of materials within glass vessels," *arXiv preprint arXiv:1708.08711*, 2017. 5

[39] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016. 5

[40] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19. 6

[41] H. Mouchere, R. Zanibbi, U. Garain, and C. Viard-Gaudin, "Advancing the state of the art for handwritten math recognition: the crohme competitions, 2011–2014," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 19, no. 2, pp. 173–189, 2016. 6

[42] M. Mahdavi, R. Zanibbi, H. Mouchère, and U. Garain, "ICDAR 2019 CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection," in *Proc. ICDAR*, 2019, to appear. 6

[43] M. Mahdavi and R. Zanibbi, "Tree-based structure recognition evaluation for math expressions: Techniques and case study." 8

[44] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognition*, vol. 71, pp. 196–206, 2017. 8