LPGA: Line-Of-Sight Parsing with Graph-based Attention for Math Formula Recognition

Mahshad Mahdavi, Michael Condon
Document and Pattern Recognition Lab
Rochester Institute of Technology
Rochester, NY, USA
mxm7832@rit.edu
mpc7497@rit.edu

Kenny Davila
Department of CSE
University at Buffalo
Buffalo, NY, USA
kennydav@buffalo.edu

Richard Zanibbi
Document and Pattern Recognition Lab
Rochester Institute of Technology
Rochester, NY, USA
rxzvcs@rit.edu

***Abstract*—We present a model for recognizing typeset math formula images from connected components or symbols. In our approach, connected components are used to construct a line-of-sight (LOS) graph. The graph is used both to reduce the search space for formula structure interpretations, and to guide a classification attention model using separate channels for inputs and their local visual context. For classification, we used visual densities with Random Forests for initial development, and then converted this to a Convolutional Neural Network (CNN) with a second branch to capture context for each input image. Formula structure is extracted as a directed spanning tree from a weighted LOS graph using Edmonds' algorithm. We obtain strong results for formulas without grids or matrices in the InftyCDB-2 dataset (90.89% from components, 93.5% from symbols). Using tools from the CROHME handwritten formula recognition competitions, we were able to compile all symbol and structure recognition errors for analysis. Our data and source code are publicly available.**

***Index Terms*—line-of-sight graph, math recognition, CNN, MST-based parsing**

I. Introduction

Mathematical notation is a key component in technical communication, and in recent years there has been growing interest in both the automatic recognition and retrieval of mathematical notation [1]. In addition to research efforts, this includes a variety of commercial systems from companies including PhotoMath, MyScript, Wiris, and WolframAlpha. The potential benefits of easy-to-enter, easy-to-modify, and easy-to-search formulas are vast: this could provide powerful new applications for technical experts, researchers, and educators. More generally, easier manipulation and access for formulas may help increase mathematical literacy in the general public (e.g., by allowing people to easily look up formulas and symbols based on their appearance).

Math encodings represent the appearance or mathematical content of formulas. As illustrated in Figure 1, the appearance and content of mathematical expressions can be defined using trees. Encodings for appearance represent a formula based on the arrangement of symbols on writing lines, which we call a Symbol Layout Tree (SLT [2]). LaTeX is essentially an SLT representation with additional formatting commands (e.g., to control sizes, spacing, and font faces). The Operator Tree (OPT) in Figure 1 represents the application of operations to operands, from the leaves to the root of the tree.

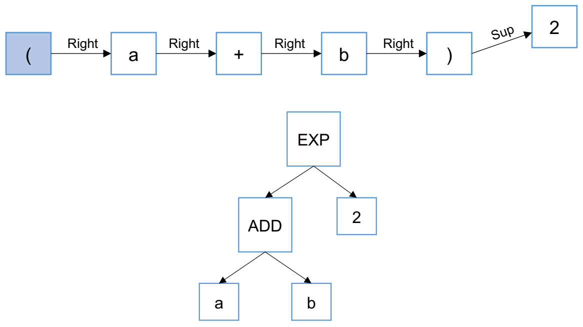

Fig. 1: Math representations: Symbol Layout Tree (SLT, top) and Operator Tree (OPT, bottom) for $(a+b)^2$

In this work, we concern ourselves with recovering visual structure (SLT) from both raster images (e.g., PNG) and vector images that provide symbol locations and identities directly (e.g., born-digital PDFs). An SLT can be transformed into an OPT using an expression grammar, but we don't consider recognizing formula semantics in this paper.

Mathematical expression recognition comprises three major tasks: detecting symbols, classifying symbols, and determining expression structure. These tasks may be solved in a sequential feed-forward manner, or in a globally integrated fashion [3], [4]. In feed-forward approaches, errors in symbol tasks are inherited by the structural analysis, and so recognition accuracy tends to decrease with increasing the expression size. Integrated approaches do not rely on previous steps, meaning that structural analysis interacts directly with symbol segmentation and symbol classification. A common approach for global solutions is applying sequence-to-sequence learning models to generate captions for formula images. These methods generate string outputs in LaTeX, using global context from attention models to avoid local ambiguity [5]. Jointly training an encoder-decoder system that maps raw images to string

We thank the **National Science Foundation (USA)** and the **Alfred P. Sloan Foundation** for their support of this work.

output is appealing, but this also makes diagnosing errors hard, as the correspondence between the output LaTeX string and the input image is difficult to determine.

In our work, we recognize formulas from connected components in images or images with provided symbols. This permits us to use evaluation tools that automatically identify errors at the level of connected components or symbols and their relationships [6], providing a concrete understanding of the obstacles and hard cases that must be overcome. For example, we saw that our ground-truth spatial relationships for punctuation needed changing to improve accuracy.

To utilize visual context in symbol and relationship classification, we apply a visual attention model organized around a line-of-sight graph over connected components or symbols [7], [8]. This disambiguates similar symbols and relationships (e.g., fraction line vs. minus), and allows a simple feed-forward system to obtain accurate results when classifying symbols *after* symbol segmentation and structure recognition.

Our Line-of-Sight Parsing with Graph-based Attention (LPGA) model parses a line-of-sight graph over connected components by (1) locating symbols, (2) extracting an SLT from line-of-sight graph over detected symbols using a maximum spanning tree, and (3) classifying symbols. This is a generalization and extension of the work by Hu using LOS graphs for parsing math that is handwritten online [7]. Here, in addition to using Random Forests with visual densities, we have also created a VGG-16-based Convolutional Neural Network with two branches to automatically learn input and context features separately. We call our new method LPGA (Line-of-sight Parsing with Graph-based Attention).

We also provide a new dataset containing formulas without grids or matrices from the InftyCDB-2 dataset [9], with annotations down to the level of connected components (InftyMCCDB-2). The LPGA source code and modified InftyCDB-2 dataset are publicly available.[1]

In the remainder of the paper, we introduce related work, define the LPGA model, present results on the InftyMCCDB-2 data set, and discuss future work.

II. Related work

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

A. Segmentation

Segmentation is a task of grouping related primitives. These primitives could be pixels from an image, or strokes from a handwritten equation. The main challenge of symbol segmentation in typeset mathematical expression images is fractured symbols whose components were split by printing and scanning noise. These cases are often hard because the appearance of the fracture might be unique or rarely seen.

Almost all state-of-the-art systems for segmentation are neural net based systems [10]–[12]. The main idea in these works is converting fully-connected layers into convolutional layers and concatenating the intermediate feature maps. Improvements in performance result either from embedding more context or generating high-resolution feature maps. Embedding context is done by employing larger kernels [13], adding a global pooling branch to extract context information like in ParseNet [14], or using Atrous Spatial Pyramid Pooling, which is a substitution for larger kernels and delivers a wider field of view at the same computational cost [10]. To generate high-resolution feature maps, techniques such as Deconvolution [12], Unpooling [15] and Dilated Convolution [16] are deployed. Some post-processing techniques such as Conditional Random Fields (CRFs) [17] or Bilateral Solvers are often employed to refine predictions near boundaries and improve segmentation accuracy.

Most methods of this type usually use complicated decoder blocks, which are computationally expensive. We believe such computation is unnecessary for the task of segmenting connected components to generate math symbols, since the nature of this problem makes it possible to utilize much simpler methods. Similar to ParseNet, we extract context from a side channel in our CNN that embeds local visual cues for each target edge between connected components. In the basic version of our system, LPGA_{RF}, we include context as visual densities using 2D grids and shape context features [7].

B. Classification

Common algorithms for symbol classification include nearest neighbor, support vector machines, random forests, hidden markov models, convolutional neural networks, and bidirectional long short-term (BLSTM) memory networks. These classifiers can be used in isolation, or combinations of classifiers may be used either in parallel or as a cascade. Nguyen et al. [18] propose a linear combination of a DMCN (Deep Maxout Convolutional Network) and BLSTM (Bidirectional Long short-term memory) networks for symbol recognition. They employ BLSTM for online features and CNN for offline features and compare them against more traditional approaches: MRF (Markov Random Field) for online and MQDF (Modified Quadratic Discriminant Function) for offline.

In typeset images, the data is inherently non-sequential with respect to time, in both the pixels that make up a symbol and the symbols themselves in the expression. Many approaches impose a spatial ordering on expression images, either a linear left to right order, or a form of two dimensional ordering. We do not commit to an order for gathering features: rather, we independently extract convolutional features around image areas predicted to be symbols by our attention graph.

C. Parsing

Parsing mathematical expressions converts input primitives (e.g., images, handwritten strokes, or symbols) to a description of formula structure. A common set of features used to represent the spatial relations between components are geometric features. Visual features have also been used recently [19], [20]. Systems that use visual features are usually built around

[1]https://www.cs.rit.edu/ dprl/software.html

neural networks. IM2TEX, inspired by a sequence-to-sequence model designed for image caption generation, directly feeds the typeset formula image into a Convolutional Network to extract a feature grid learned by the network [5]. The feature grid is passed through an LSTM to generate candidate LaTeX tokens. The token list is then given to a decoder to build the expression string. Results show their system produces LaTeX strings that when rendered exactly match the input image for 79.88% of the IM2LATEX-100K test dataset [5]. Their approach is interesting in that it is trained using only images and LaTeX strings, and produces a LaTeX string as output. Diagnosing error is difficult when expressions are incorrect, as the correspondence between the string and input image is not simple to determine.

In Zhang et al. [21], a multi scale attention model is used in the encoder with the purpose of preserving details, which improves recognition of handwritten inputs. The encoder is made of two branches, i.e., except for the main branch which produces low-resolution annotations, another feature map is extracted before the last pooling layer which has higher resolution.

Many systems do not use a single algorithm to go from images to a fully recognized expression, instead they look at the problems individually and combine results from subtasks. Syntactic methods combine recognition modules using an expression grammar, which both defines legal expressions and drives the search for an interpretation by the parser. In state-of-the-art systems, production rules are probabilistic (e.g., using Stochastic Context-Free Grammars (SCFGs) [22]). Grammar-based systems continue to produce competitive results [23], but grammars require manual construction, and expressions not captured by a grammar produce errors.

III. LPGA Model

Recovering a formula SLT from an image can be posed as a graph search problem [24]. In our case, we attempt to extract an SLT containing symbols and their associated spatial relationships from a weighted graph defined over connected components (in raster images) or symbols (for vector images). Unlike grammar-based techniques such as Stochastic Context-Free Grammars (SCFGs) [4], graph-based parsing does not require an expression grammar; the language model consists of only node and edge labels. Parsing a graph involves identifying a sub-graph with minimal cost or maximum probability subject to certain constraints. For math recognition, the final subgraph is usually a Symbol Layout Tree.

Ideally, for graph-based parsing the search graph contains edges representing all relationships and symbols (i.e., perfect recall), with few extraneous edges increasing the search space size (i.e., high precision). Previously, Line-Of-Sight (LOS) graphs have been used to parse handwritten formulae; for the CROHME handwritten math recognition benchmark, they provide perfect recall for stroke pairs belonging to symbols, and can represent SLTs for roughly 98% of CROHME formulae, which is much higher than for Time-series, Minimum Spanning Tree, Delaunay, and k-NN graphs for $k \in \{1, 2, \ldots, 6\}$ [8]. On average, the number of edges in the LOS graphs was 3.3 times the number of strokes, which is much smaller than the quadratic number of edges in a complete graph providing perfect SLT recall ($n(n-1)/2$ for n strokes).

We parse LOS graphs to recognize formulas in images, rather than in online handwritten strokes. As shown in Figure 2, our **LPGA** (Line-of-Sight Parsing with Graph-based Attention) model first generates a LOS graph over connected components. A binary classifier then identifies which directed LOS edges correspond to components that should be merged into symbols. Symbols that can see each other are connected in a second LOS graph. Edges are classified into nine spatial relationships, including 'NoRelation' (see previous Section). A classifier is used to determine the highest relationship probability for each edge; NoRelation is filtered to prevent extracting a *forest* of disconnected subtrees, which is invalid for an SLT. Edmonds' algorithm [25] then extracts a directed maximum spanning tree from the relationship probabilities on the symbol LOS graph edges, producing an SLT. The final step is to classify the symbols in the extracted SLT (for space, this is not shown in Figure 2). Symbol classification is performed on segmented symbols using only visual features.

Line-of-Sight Graph Modifications. Some structures in mathematical expressions directly impact the line-of-sight graph construction. A common problem is that baseline punctuation such as commas, periods, and dots are blocked by subscripted expressions (see Figure 6). We modify LOS graphs to check whether a given symbol has a much smaller blocked symbol in roughly the subscript region. To restrict where baseline punctuation is looked for, we use an angle range of 310-360 degrees in which the child symbol must be at least 3.5 times smaller than the parent. If so, then an edge is added from the symbol to its 'blocked' symbol in the subscript region. The second modification reduces the amount of blocking caused by subscript and superscript symbols. We define 'transparent edges' on the symbol bounding boxes, where the angle blocked by a child symbol is reduced by shrinking its apparent bounding box size.

Geometric Features. Geometric features are used for classification of split/merge relationships between connected component pairs, and symbol relationships between symbol pairs. These features include distance measures, area overlaps and differences, size ratios, and angles. For pairs of connected components or symbols a common set of features are distances and differences based on the bounding boxes around each stroke/symbol. These distances include: distance between center points, difference in vertical position of bounding box tops and bottoms, difference in horizontal positions of left and right edges, difference in area, and amount of overlapping area. In particular, we have found that geometric features work well when trying to discern spatial relationship between symbols.

Classification Models. In the first version of our system, LPGA_{RF}, visual density features were captured using grids and shape contexts [26]. The resulting histograms for the input and its local context were combined with geometric features. The local context was defined using a neighborhood size fixed

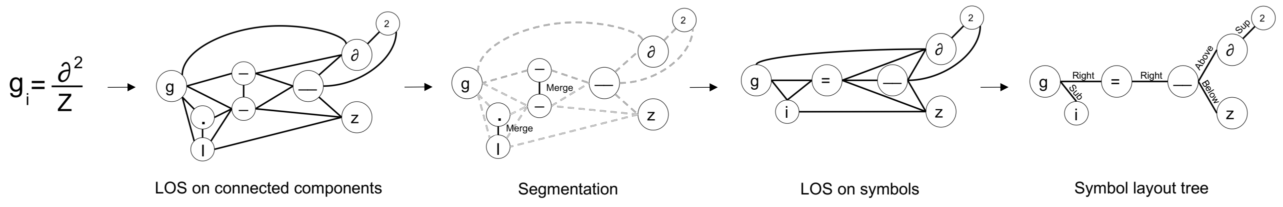

Fig. 2: Recognizing formula structure from connected components (CCs) in an image. CCs share an edge if an unobstructed line can be drawn from one bounding box center to a point on the convex hull of the other. After segmentation, symbols are connected in a second LOS graph, and the spatial relationships represented by edges are classified. Edmonds' algorithm then selects a maximum spanning tree to produce a Symbol Layout Tree (SLT).

relative to the input. Random Forests are used for classification of symbols and edges.

Features are extracted only from image regions containing components or symbols sharing an edge in an LOS graph, and for symbol nodes in the symbol LOS graph. This means instead of exhaustively searching the input image to find salient areas, we have a predefined map based on edges and nodes of the LOS graph (see Figure 3). This allows us to meaningfully capture visual context without using a probabilistic/sequential attention model. We instead use hyper-parameters defining the amount of visual context to use in each classifier type (binary segmentation, relationship classification, and symbol classification).

In the second version of our system we use Convolutional Neural Networks for classification of LOS edges and symbols (LPGA$_{\text{CNN}}$). Visual features for inputs are learned and extracted automatically by a two channel multi-layer convolutional neural network, using the standard VGG-16 architecture in each channel [27]. Thus, for each target image, a parallel branch provides contextual information around the target for better decision making (see Figure 3). Features from these two channels are concatenated before being passed through a softmax layer. For segmenting connected components and classifying spatial relationships, the geometric features described above are added directly to the final feature vector before classification. All features values are normalized between [-1,1].

We use the same architecture for classifying pairs of connected components as merge/split (see Figure 3), classifying relationships between pairs of symbols, and classifying individual symbols. They differ only in that symbol classification takes a single symbol as input rather than a pair of components/symbols, and so the associated geometric features for those pairs are not added before the final dense layer. Figure 3 shows our CNN classifier for segmenting connected components into symbols, by classifying each edge in a LOS graph over connected components as 'merge' or 'split' (see Figure 2). In Figure 3, the input is the pair of connected components in the equals sign (=). If we were classifying the '=,' processing would stop at the first dense layer.

Implementation and Training. The CNN specifications are explained in [27] and visualized in Figure 3. We remove the fully connected layers and add a Global average pooling layer followed by a Dense layer with a drop out rate of 0.5. We freeze the first four blocks with the ImageNet pre-trained weights and train the top layers. Non-trainable layers are grayed out in Figure 3. We train the two branches jointly by updating all the trainable weights after passing each batch, and force the network to use the visual clues from the context channel to classify the target input. The model has a total of 30.48 million parameters, of which 15.21 million are trainable. We use an Adam optimizer to learn the parameters. The batch

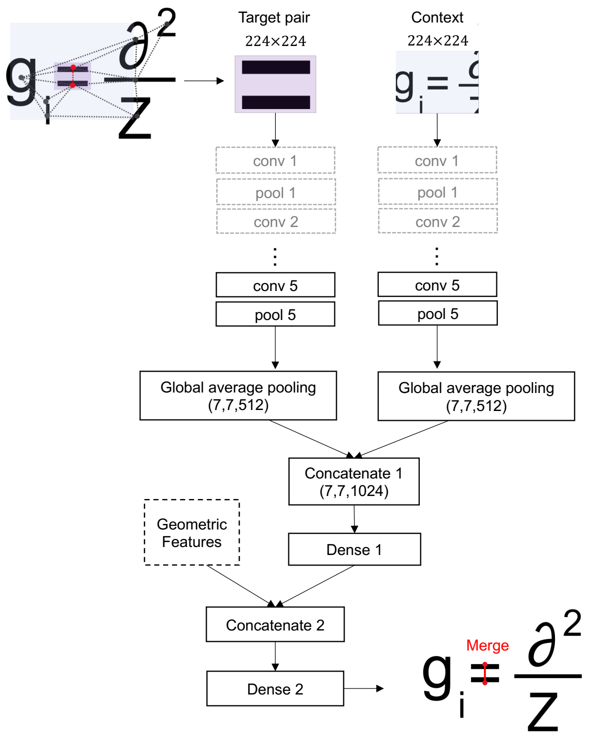

Fig. 3: LPGA$_{\text{CNN}}$ classification architecture. In this example, the red edge between connected components in the equals sign (=) is being classified for symbol segmentation (see Figure 2). Two independent branches based on VGG-16 blocks represent the input (left side) and the attention context (right side). Geometric features are added before the final dense layers.

size was set to 32, momentum to 0.9 and the learning rate was initially set to 10^{-3}, 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 Keras [28] and experiments were run on an 8GB Nvidia 1080 GPU.

The implementation used for random forests is from the python scikit-learn library. We set the parameters at: 50 trees, maximum tree depth of 40, 30 randomly sampled features at each node during training, and the Gini measure was used to select split points. These 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.

IV. The InftyMCCDB-2 Dataset

For our experiments we use InftyMCCDB-2, a modified version of InftyCDB-2 [9] which contains mathematical expressions from scanned article pages. The dataset has 21,056 mathematical expressions. We remove formulas with matrices and grids, leaving 19,381 formulas. The dataset includes 213 symbol classes, and is split into two sets: training (12551 images), and testing (6830 images) with approximately the same distribution of symbol classes and relation classes. The expressions range in size from a single symbol to more than 75 symbols, with an average of 7.33 symbols per expression.

The original InftyCDB-2 provides ground truth at the symbol level. We extracted connected component bounding boxes, and generated new ground truth for each image using a labeled adjacency matrix ('label graph') representation [6]. When generating the CCs, we ensure that each connected component belongs to at most one symbol, based on character bounding box information. Table III indicates detection results on individual symbols. A label graph file stores connected components with individual identifiers, groupings of components into symbols with their labels, and finally directed Symbol Layout Tree edges.

There are seven spatial relationships in InftyCDB-2: horizontal (HORIZONTAL), right/left superscript (RSUP and LSUP), right/left subscript (RSUB and LSUB), above (UPPER) and below (UNDER). We add an eighth spatial relation for baseline punctuation (PUNC).

Punctuation Representation. As seen in Figure 4, punctuation symbols are spatially more similar to subscripted symbols than horizontally adjacent symbols. Having punctuation symbols on the baseline separated from symbols on the main baseline, and placed in their own nested region relative to their parent symbol allows horizontal relationships to be represented in a more consistent manner, and punctuation to be associated with subexpressions more accurately. Baseline punctuation is given its own relation class, *punctuation* (PUNC, see Fig. 4). PUNC relationships can be easily converted back to HORIZONTAL relationships using a simple graph rewriting rule.

V. System Design Experiments

A series of experiments were conducted to refine our CNN classification model for segmentation, symbol relationships,

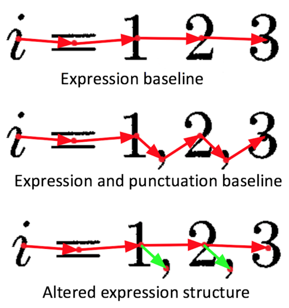
Expression baseline
Expression and punctuation baseline
Altered expression structure

Fig. 4: Modifying SLTs for punctuation. As shown in the middle image, punctuation is represented using horizontal relationships (red edges) despite the marked shift in vertical positions. To avoid inconsistent 'horizontal' relationships, we define a punctuation relation (PUNC, shown in green).

and symbols. For each set of experiments, parameters and structure designs which give the best result in a 5-fold cross validation were used for testing.

Context. These experiments aimed to study context in feature extraction. Experiments showed that having context is helpful in all three tasks. It is expected that contextual information will be beneficial for labelling edges, but it is also useful for symbol recognition. Having a more global view resolves class ambiguities for visually similar classes as shown in Table I. Symbol classification accuracy on the test set improves from 98.13% to 99.43% when having a second branch for context.

How much context should we use? Previous experiments for $\mathrm{LPGA_{RF}}$ found that increasing histogram size to capture more visual context from surrounding symbols improves classification accuracy. We did a grid search to optimize the radius of the context window in the second channel for each task. We find a radius factor (α), which is multiplied with original target radius and produce the context radius: $r_{\text{context}} = \alpha \times r_{\text{target}}$. A radius factor of 1.75, 1.5 and 4.0 produced the best results for segmentation, parsing and classification respectively. It is worth noting that the input includes two connected components or symbols for segmentation and parsing and only a single target symbol for classification (r_{target} is larger for the first two tasks); so the chosen radius for all three tasks provide similar contextual information.

ALL vs ELSE in context channel. The next set of experiments, studied whether we should keep the target in the context window (ALL) or just pass everything around it (ELSE). We hypothesize that ALL should perform better, as at some depth in the context channel the field of view will capture the target, and since the two channels are mutually trained, it hopefully will learn the target shape. Hence, by not removing the target from the context channel, the model can potentially capture some relative positions. Experiments show little difference between these two situations, e.g., less than 0.5% for segmentation, so we decided to pass the entire image region into the context branch.

Early Fusion vs Late Fusion. We also tested alternative

TABLE I: Effect of visual context attention on classification of similar symbols. Shown are the top-5 most frequent confusions for visually similar classes before using context, and then after.

		Errors	
Ground Truth	**Prediction**	**No Att.**	**w. Att.**
overline (e.g., $\overline{x}$)	minus	192	27
fractionline	overline	122	10
minus	overline	94	9
cdots ($\cdots$)	ldots ($\ldots$)	21	8
letter l	one (1)	18	10

ways of representing contextual information. We initially did source separation in the inputs by having the target in one channel, and context in the other channel (early fusion). This works well when using visual density features in LPGA$_{\mathrm{RF}}$. Instead of having one visual density histogram, there are three histograms for parent, child and context. In the convolutional layers of LPGA$_{\mathrm{CNN}}$, the three channels for parent, child and context will collapse into one 2D feature map once it convolves with the first kernel. The kernel moves along the height and width with the defined stride and its depth is the same as the inputs, e.g., in first layer kernels have depth of three if the model accepts RGB images. In other words, RGB channels of an input image are merged very early and basically no individual weight is trained for target and context individually as we hoped.

To have separate weights for the input and context, and in order to train both mutually, we decided to have a separate branch for context and merge the branches later in the pipeline. This allows the network to see the context while focusing more on the target. It worth mentioning that another advantage of late fusion is faster convergence during training.

We also explored a three-branch structure in which the parent and child was fed to the system separately and a third branch to capture context, but the performance deteriorates compared to having parent and child as target in one channel. That could happen due to the fact that we are freezing weights in the first four convolutional blocks in all three branches, and thereby not back propagating through all the layers during training. Without this, training would not converge.

Multi-Resolution Context. Low resolution context improves the result when using visual density in LPGA$_{\mathrm{RF}}$. That encouraged us to investigate different resolutions for the context channel. Experimental results showed that coarser context lowers the accuracy. We investigate three different resolution, 224 pixels height and width which is the default size for VGG16, 112×112 pixel resolution and 48×48. Decreasing the context input resolution in symbol classification task with a late fusion structure and context radius factor of 4 decreases the accuracy from 99.34(%) into 99.03(%) for halving the input size and 98.24(%) for the smallest inputs (48×48).

Spatial Features on Side Channel. For defining the relation between connected components in segmentation or symbols in parsing, relative position of parent and child plays a key role. We studied whether embedding the structural information extracted from the input graph into classification is useful. We concatenate the spatial features extracted from the node pairs in LOS graphs with the final feature vector before passing it through the softmax layer. Feature vectors are normalized using a $\tanh$ activation function before concatenation meaning the last ReLu function is also replaced with $\tanh$ for domain adaptation. The result shows geometric features can improve the performance of relation classifiers, e.g. from 98.69% to 99.34% for the segmentation task.

VI. Experiments on InftyMCCDB-2

In this section, we present results for LPGA using random forests vs. CNNs for classification on the InftyMCCDB-2 dataset. Results are compiled using the LgEval library [6] created for the CROHME competitions. Our evaluation metrics are recognition rates for formulas, and F-scores for symbol and relationship detection and classification.

Expression-Level Results. The main experimental results show that Line-Of-Sight Parsing with Graph-based Attention is effective for scanned typeset math expressions (see Table II). Results are presented under two conditions: from connected components, and from given symbols (i.e., where segmentation and classification rates are 100%). From connected components, our system recognizes 90.89% of expressions perfectly, i.e., no error in character recognition or structure analysis. For comparison, the reported expression rate for the INFTY system is 89.6% [29] on the *original* Infty-CDB2 dataset (i.e., including formulas with matrices and grids). Examples of correctly recognized formula images from the dataset are shown in Figure 5.

Table II also shows the percentage of test formulas where connected components are correctly merged into symbols, and the number of formulas with correctly segmented and classified symbols. Similar results are reported for formulas without relationship errors in Table II. A correctly detected relationship requires a pair of valid symbols to share or not share a relationship as indicated in ground truth. Formulas have correct structure if symbols and relationships are detected correctly, ignoring their classification. Finally, in the rightmost column we see the expression rate, where formulas have both valid structure and symbol/relationship classification.

Using our CNNs rather than random forests improves the expression rate for raster images, but not for parsing from given symbols. The biggest improvement produced by our CNNs is for symbol classification, where the automatically inferred convolutional features seem to be more effective than the visual density histograms used in LPGA$_{\mathrm{RF}}$. Our CNN-based relationship classifier performs just slightly less well than the random forest model. It is possible that the separation of each pair of components/symbols in separate channels might be producing this (very small) improvement. However, the CNN models provide more opportunity to produce globally optimized end-to-end trainable systems in the future.

In terms of memory and run time, RF classifiers require 440M space and CNN model size is 700M. The execution time is 1.38 seconds per sample for LPGA$_{\mathrm{CNN}}$ and 1.62 seconds per sample for LPGA$_{\mathrm{RF}}$, on average.

TABLE II: InftyMCCDB-2 test set results for correct symbol/relationship locations (*Detection*), correct symbol/relationship classes (*Det.+Class*), unlabeled formula SLT structure, and structure with correct labels (*Str.+Class*). Percentage of correct formulas are shown.

	Symbols		Relationships		Formulas	
	Detection	Det.+Class	Detection	Det.+Class	Structure	Str.+Class
CC Input						
LPGA$_{\text{RF}}$	97.18	92.80	**93.81**	**93.25**	**93.81**	90.06
LPGA$_{\text{CNN}}$	**97.29**	**95.17**	93.37	92.59	93.37	**90.89**
Symbol Input						
LPGA$_{\text{RF}}$	–	–	**94.36**	**93.50**	**94.36**	**93.50**
LPGA$_{\text{CNN}}$	–	–	94.28	93.00	94.28	93.00

Symbol Segmentation. Table III indicates detection results for individual symbols. In LPGA$_{\text{RF}}$, the most frequent failure cases are over segmentation of lower case i, and double prime. In both cases the individual connected components look like other symbol classes. These cases are segmented correctly with the aid of context in LPGA$_{\text{CNN}}$. For the CNN, undersegmentation occurs when two symbols are close to each other, which probably is a consequence of missing details in pooling layers. Figure 6 shows some of these segmentation errors. The most frequent error is merging h with overline. Another common error is merging 1 with a fraction line. Interestingly, Table 5 in Mouchère et al. [6] shows that this also is the most common error for handwritten math expressions in previous CROHME competitions.

To avoid such confusions, we plan to either extract multi resolution-feature maps from different convolution blocks to compensate for fine detailed dropped by pooling layers, or decrease the number of pooling layers in the context branch.

Symbol Classification. Shape-based classification leads to the problem of trying to distinguish symbol classes with identical or similar shapes. In both the CNN and RF versions of LPGA, use of visual context that improves symbol classification, as seen for LPGA$_{\text{CNN}}$ in Table I.

From Table III, the classifier in LPGA$_{\text{CNN}}$ performs slightly better. Figure 6 shows some of the most common symbol recognition errors for LPGA$_{\text{CNN}}$. In the first row, the calligraphic 'F' is recognized as '$=$'. The confusion between 'α' and 'a' is not completely solved with context, since their spatial position does not differ as much as other visually similar pairs such as fraction line & minus, overline & minus, etc.

Expression Parsing. Before adding the PUNC spatial relationship, the two main recognition errors for punctuation were adding them to the end of a subscripted baseline, or incorrectly classifying the relation as a subscript instead. After adding the PUNC relationship, the frequency of these errors were reduced, and more relationships are detected and correctly classified, increasing the expression rate by roughly 2% for the RF and CNN-based models.

A relationship class confusion matrix shows that most errors are missed relationships, with missed PUNC relationships being the most frequent. The most common errors are missing PUNC edges between a and comma, and x and comma. Although our modified LOS construction tries to avoid problems with subscripted expressions blocking the line-of-sight between a symbol and its associated comma or other punctuation, this still occurs with large subscripts (see Figure 6).

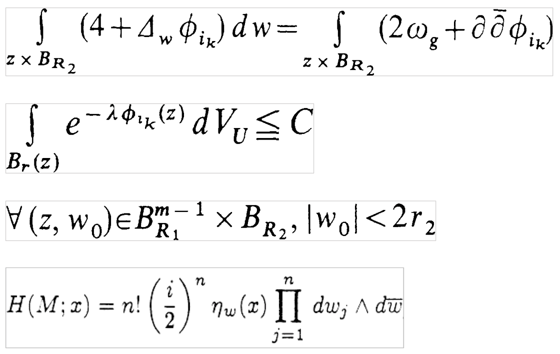
Fig. 5: Images correctly recognized by LPGA$_{\text{CNN}}$.

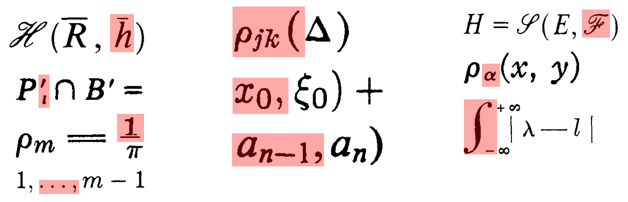
(a) Segmentation (b) Parsing (c) Symbols

Fig. 6: LPA$_{\text{CNN}}$ errors. (a) overline merged with h, prime merged with i, 1 merged with fraction line, comma merged with dots. (b) Missing LOS edges blocked by subscripts; missing HOR relation between ρ and (, missing PUNC relation between comma and x, and between comma and a. (c) Calligraphic F classified as equal, α confused with a, and integral classified as f.

VII. Conclusion and Future Work

We have presented a novel approach for recognizing typeset formula images, in which we extract features using a simple attention model organized around a line-of-sight graph over connected components. Constraining the search space from pixels to nodes and edges of the attention graph appears to

TABLE III: InftyMCCDB-2 Results for individual symbols and spatial relationships recognized from connected components.

	Symbol Detection			Symbol Detection + Class		
	Precision	Recall	F-score	Precision	Recall	F-score
LPGA_{RF}	**99.24**	99.44	99.34	98.41	98.61	98.51
LPGA_{CNN}	**99.24**	**99.45**	**99.35**	**98.84**	**99.05**	**98.95**

	Relation Detection			Relation Detection + Class		
	Precision	Recall	F-score	Precision	Recall	F-score
LPGA_{RF}	97.43	**98.23**	97.83	97.17	97.96	97.56
LPGA_{CNN}	**97.71**	**98.23**	**97.97**	**97.48**	**98.00**	**97.74**

be effective for scanned typeset expressions, and we obtain strong results using relatively simple classifiers and maximum spanning tree extraction. Contextual features improved classification accuracy for merging connected components, spatial relationships, and symbols.

The current system performs recognition tasks in isolation using a predefined attention. A promising direction for future work is training the classifiers jointly on all edges and nodes in graphs over connected components. This could solve errors that happen locally by providing a global view of the entire expression. Another opportunity would be using multi-resolution feature maps extracted from different convolution blocks, in order to avoid losing fine details dropped by pooling layers. Finally, grids, matrices, and tables may be represented as trees, and we believe that we can generalize our technique to handle these simply by expanding the set of spatial relationships [30].

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.

[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.

[3] 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.

[4] 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.

[5] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," *arXiv preprint arXiv:1609.04938*, 2016.

[6] 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.

[7] 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.

[8] ——, "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.

[9] M. Suzuki, S. Uchida, and A. Nomura, "A ground-truthed mathematical character and symbol image database," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. IEEE, 2005, pp. 675–679.

[10] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[11] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[12] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[13] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel mattersimprove semantic segmentation by global convolutional network," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 1743–1751.

[14] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[15] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[16] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[18] H. Dai Nguyen, A. D. Le, and M. Nakagawa, "Deep neural networks for recognizing online handwritten mathematical symbols," in *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*. IEEE, 2015, pp. 121–125.

[19] 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.

[20] 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.

[21] J. Zhang, J. Du, and L. Dai, "Multi-scale attention with dense encoder for handwritten mathematical expression recognition," *arXiv preprint arXiv:1801.03530*, 2018.

[22] 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.

[23] 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.

[24] 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.

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

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

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] F. Chollet *et al.*, "Keras," 2015.

[29] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "Infty: an integrated ocr system for mathematical documents," in *Proceedings of the 2003 ACM symposium on Document engineering*. ACM, 2003, pp. 95–104.

[30] R. Zanibbi, K. Davila, A. Kane, and F. W. Tompa, "Multi-stage math formula search: Using appearance-based similarity metrics at scale," in *Proc. Int'l ACM SIGIR Conference on Research and Development in Information Retrieval*, 2016, pp. 145–154.